Learn representations in the presence of segmentation label noises

Jihong Ju
Computer Vision Lab, TU Delft
Mekelweg 4, 2628 CD Delft
j.ju@student.tudelft.nl

August 27, 2017

Abstract

Training data for segmentation tasks are often available only on a small scale. Transferring learned representations from pre-trained classification models is therefore widely adopted by convolutional neural networks for semantic segmentation. In domains where the representations from the classification models are not directly applicable, we propose to train representations with segmentation datasets that potentially contains label errors. Our experiments demonstrate that label errors, such as mislabeled segments and missing segmentations, have negative influences to the learned representations. To alleviate the negative effects of object mislabelling, we propose to discard the object labels and instead train foreground/background segmentation. The learned representations with binary segmentation achieve a fine-tuning performance comparable to the representations learned with "gold" standard segmentations. In the existence of missing segmentations, a sigmoid loss for the background class is proposed to achieve high recall while keeping the precision better than simply weighting the classes. The proposed class dependent, sigmoid loss achieves both better pre-training performance and better fine-tuning performance than the weighting the classes in the presence of missing segmentations. To summerize, we propose to learn representations with foreground/background segmentation and with a sigmoid loss for the background class when there exist missing segmentations for objects.

1 Introduction

The often limited availability of training samples motivates most state-or-the-art deep learning based segmentation models [26, 2, 14] to transfer convolutional neural network (CNN) models [19, 36, 38, 15] trained on a subset of images from ImageNet. The difficulty of obtaining manual segmentations is natural because it costs much more efforts for people to segment than to classify an image. One of the largest segmentation datasets, Microsoft COCO2014 [24], contains 123,287 images of 80 object categories. As a comparison, a well-known successful task for convolutional neural networks, object recognition on the ILSRVC dataset[34], has around 1.2 million images for 1000 categories to train. Transferring weights from the pre-trained ImageNet models can provide a segmentation performance boost in the limitation of lacking training samples, as reported in [26] and adopted by [2, 14]. But the pre-trained ImageNet models are originally designed for object recognition problems, which can cause more problems than it solves.

In practice, it can be challenging to employ representations from the ImageNet CNN models directly for segmentation. Firstly, the object recognition models pursue features invariance to better capture semantics regardless the variations in objects. The result translation invariant and resolution-reduced features reduce the localization accuracy which is not essential for object recognition but is critical for object segmentation. [43, 2] Secondly, the ImageNet models were originally trained with natural images at relatively low resolution. However, images to be segmented may (1) have a third dimension (3D images

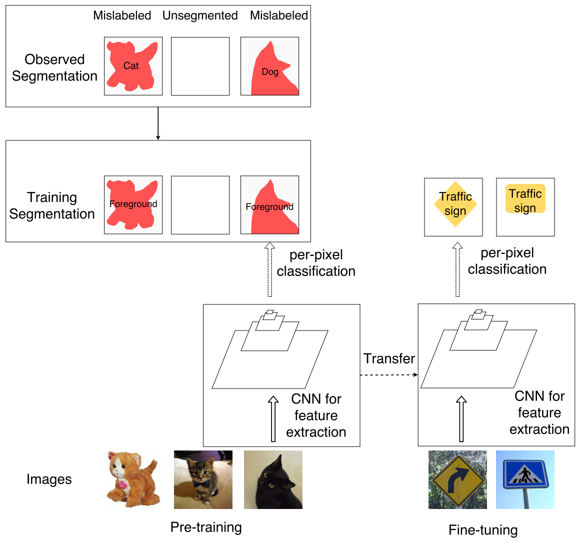

Figure 1: Learning representations with segmentation datasets that potentially contains mislabeled objects and missing segmentation. We propose (1) to train with only segmentations instead of labeled segmentations, (2) to apply a sigmoid loss for the background class. The learned representations are then used as weights initialization and fine-tuned with a small set of true labels in the domain of interest.

like CT scans and MRI scans), (2) contain extra channels (RGB-D images), (3) be non-natural, such as aerial images and medical images. These issues prevent transferring representations of the ImageNet models from improving segmentation performance significantly. In this case, it can be beneficial to retrain the pre-trained ImageNet models with segmentation datasets for fine-grained cues about boundaries in the domain.

The segmentation datasets for pre-training representations may contain label errors. The use of the crowd-sourcing platform like Mechanical Turk is common nowadays to collect annotations on a large-scale. It is natural for crowd-sourcing workers to make mistakes as a result of lack of expertise, inherent ambiguity of tasks or unconscious bias. Enormous efforts are required, according to [24, 10], to ensure the correctness of segmentations. In addition, automated labels other than the manual ones may be freely available for particular tasks. For example, segmentations of road and buildings for aerial images can be derived from digital maps, like OpenStreetMap, by aligning images to maps. However, segmentations constructed in this way suffer from incompleteness as well as registration problems [28]. Ideally, label errors in segmentations should not significantly affect the learned representations and its transferability to other datasets.

Label errors of different kinds can exist in segmentation labels. We consider mislabelling errors occurred to the whole segment instead of individual pixels, assuming the outline of objects is always correct. This is based on the observations that most objects in natural images have visually clear borders, and it may be untrue in some cases, for example, context segmentations[29]. In particular, we consider three types of label errors: inexhaustive segmentation, objects mislabelling, and false positive segmentations. **Objects mislabelling** from one category to another exist occasionally even for well-annotated datasets. For example, the Microsoft COCO dataset [24] contains some mislabeled cats and dogs even though annotators were asked to segment only one category at a time; **Inexhaustive segmentation** means that there exist objects left unsegmented. A typical scenario where incomplete segmentation emerges is to segment images containing massive amounts of objects of the same kind, e.g., a flock of sheep or a pile of products; **False positive segmentation** denotes that semantically meaningful objects from an undefined category are wrongly segmented, as objects of interest. For instance, a dataset may contain segmentations for toy cats, labeled cats, given that toy is not one of the categories of interest and cat is. We report in this work that objects mislabelling and inexhaustive segmentation both have a negative influence on the learned representations, whereas the false positive segmentation has little effects.

If negative influences to the learned representations introduced by label noises are remarkable, methods to compensate the errors become necessary. To overcome the negative influence of objects mislabelling, we propose to group all object categories into one foreground class and train representations by learning to segment foreground and background. Incorrect foreground labels can be considered as precise but inaccurate measurements of object class, whereas the label "foreground" is accurate but imprecise for segmented objects. Grouping object categories can be regarded as converting precise but potentially inaccurate labels to accurate but imprecise labels. We argue that learning representations do not require as precise supervision as learning classifiers. As a matter of fact, how well the learned representations transfer to another dataset is inversely correlated to its dependence of specific categories [42]. In addition, Jain et al. [18] demonstrated a fully convolutional network trained on over one million images to for binary segmentation generalizes well to thousands of unseen object categories. This observation indicates that a convolutional network can learn generic knowledge about object boundaries if it can segment foreground and background for a wide range of categories sufficiently well. Therefore, we propose to learn representations by foreground/background segmentation instead of per-class segmentation.

If we consider datasets contained missing segmentations, the problem becomes similar to a so-called *positive and unlabelled learning* (PU learning) setup [23]. In the positive and unlabeled learning setup, the training dataset has two sets of examples: a *positive (P) set*, containing only positive examples, and an *unlabeled (U) set*, containing a mix of positive or negative examples. Semi-supervised learning techniques are not applicable in this scenario as a result of the absence of negative training samples. The set of background pixels mixed with unsegmented object pixels, in general, fulfills this property. In an incompletely segmented dataset, pixels of the segmented objects form the P set, and the rest pixels construct the U set. Training with a segmentation dataset with incomplete segmentations is therefore similar to a learning problem with only positive examples and unlabeled examples. In this work, we treat the unlabeled set as a set of examples with noisy negative labels and propose to use the sigmoid loss for the negative class.

To summarize, the main contributions of this work are:

1. Apart from the negative influence on classification accuracy, we present that label errors also have negative influences on learning representations.

2. Instead of by training per-class segmentation, we propose to learn representations by training foreground/background segmentations when the segmentation are heavily mislabeled.

3. When training CNN models with positive and unlabeled examples, we propose a class-dependent sigmoid loss to balance precision and recall more effectively than weighting losses for different classes.

The rest of this thesis is organized as follows: In the next section, we summarize related works. In Section 3 we formulate the model for segmentation model and learning with positive and unlabeled data. We introduce the class-dependent, sigmoid loss for the negative class for deep learning with positive and unlabeled examples in Section 4. Experiments in Section 5.3 are designed to investigate the influences of objects mislabeling, inexhaustive segmentations, and false positive segmentations independently, and validate whether our proposed methods can alleviate the negative influences. The proposed sigmoid loss is evaluated, compared to weighting classes, in simulated PU learning setups in Section 5.1. Discussions are presented in Section 6 and conclusions are summarized in Section 7.

2 Related works

Transfer Learning Weights of convolutional neural networks were proved "transferable" not only to another dataset, for example, interstitial lung disease (ILD) classification [35], but also to other applications like object detection [11], and semantic segmentation[26]. Transferable means initializing the model with weights from a pre-trained CNN model results in an improvement of the model performance compared to the random initialization. [30] Yosinski et al. [42] discovered that the transferability of features is correlated with feature generality, i.e., how much the feature depends on a particular category. They also reported the weights from low-level layers of CNN models are well transferable to dissimilar categories, for example, from natural objects to human-made objects. Because features are transferable regardless the exact categories they are trained with, we argue that binarizing or categorizing the pre-training classes is expected to have no significant influence to the transferability of the result pre-trained models.

Apart from the supervised pre-training, one can also perform unsupervised learning to obtain pre-trained features in the absence of labeled training data, typically with auto-encoders [40, 27], deep belief networks [16, 21].

Though a few studies [9, 8, 1] discussed the advantage of unsupervised pre-trained features compared to random weights initialization, the difference between the two has been diminished ever since the arises of modern initialization strategies, namely Xavier initialization [12] and its variants. We used random weights initialization as the lower baseline for pre-training with noisy labels. Representations learned with supervision in the presence of label noises should at least outperform random weights because noisy information should be still better than no information.

Deep Learning with Noisy Labels The impact of randomly flipped labels on classification performance has been investigated by [37, 31] for convolutional neural networks. They both reported decreases in classification performance as the proportion of flipped labels increases for a fixed number of training samples. On the other hand, Rolnick et al. [33] argued that deep neural networks can learn robustly from noisy datasets as long as appropriate choices of hyperparameters were made. They studied the effect of label noise by diluting correct labels with errored labels instead of corrupting correct labels with errored ones and argued that collecting more labels is of more importance than correcting the obtained labels. None of these studies explored the influence of label noises on feature transferability. To the best of our knowledge, we are the first research to investigate representations robustness to label noises.

To alleviate the negative effects on classification performance introduced by errored labels, a few methods were proposed for deep neural network models. Sukhbaatar et al. [37] introduced a linear noise layer on top of the model output, and Patrini et al. [31] proposed two forms of loss correction concerning the label observation bias. Xiao et al. [41] integrated a probabilistic graphic model to an end-to-end deep learning system to predict the observed labels and to correct the observed labels. Additionally, Reed & Lee [32] proposed a bootstrapping loss to emphasize *perceptual consistency* when learning in the presence incomplete and errored labels. All these methods are proposed to solve label errors from any class to any class but often have the capability of solving specific errors from one class to another. In our problem of learning with only positive and unlabeled data, the unlabeled data can be treated as a set of examples assigned with correct negative labels and incorrect negative labels. The problem then converts to learning in the presence of label errors from positive to negative but not from negative to positive. We modified the bootstrapping loss to interpret the prior knowledge that positive labels are reliable, and set a benchmark in the experiments for the state-of-the-art methods.

Positive and Unlabeled Learning Traditional methods to learn with only positive samples and unlabeled samples for text classification [25, 23] often follow a two-step strategy: (1) first identifying a set of reliable negative samples (RN set) from U set and (2) then iteratively build a set of classifiers with RN set and P set, while updating the RN set with a selected classifier. Methods following this two step strategy do not extend well to deep learning models because it would take tremendously longer time to iteratively train a sequence of deep learning models than to train a sequence of naïve Bayesian (NB) models or supported vector machines (SVMs). For this reason, we do not consider training deep neural networks following this two-step strategy in this work.

Alternatively, one can treat all unlabeled examples as negative, and weight the losses for positive and negative examples differently [22]. Under the assumption that which positive examples are selected to be labeled is completely at random, i.e., independent of the input features, Elkan & Noto [7] proved that the probability for an object of being observed as positive differs from the probability of being truely positive by a constant factor. They also observed that a classifier trained on positive and unlabeled examples predicts probabilities that differ by only a constant factor from the true conditional probabilities of being positive. These two works considered only binary classification. We provide an extension of binary PU learning to multiclass PU learning where examples from multiple relevant classes are partially unlabeled and mixed with examples for the non-relevant class.

The often used logistic loss for neural networks grows to infinity as the confidence of wrong prediction increases to one. This can be a problem for class-weighed loss: the superfluous penalty for confident, positive predictions, i.e., samples far from the decision boundary have a large influence on the final solution. [39] Du et al. [3] illustrated that the logistic loss and the hinge loss perform worse than the ramp loss in the PU classification setting due to their superfluous penalty for confident predictions. The non-convex Ramp loss [4] and a convex double hinge-loss [3] were proposed separately to learn from positive and unlabeled data by Du et al. But neither of the two losses are continuous, which is problematic for a gradient based optimization so that we turns to a continuous altenative of the ramp loss, the sigmoid loss.

Tax & Wang [39] proposed to use the sigmoid loss for the positive class to retrieve relevant objects from a large, non-relevant objects dominant dataset, with only poorly labeled relevant objects. PU learning is happening on an opposite side of this retrieving problem: the positive examples are labeled reliably and the unlabeled examples can be considered as poorly labeled negative examples. So we proposed to use the sigmoid loss[39] for the negative class to alleviate the superfluous punishment for confident, positive predictions.

3 Problem Formulation

In this section, we formulate the model for semantic segmentation and learning with positive and unlabeled data (PU Learning).

Model for semantic segmentation A deep learning model for semantic segmentation normally consists of two principal functions: a CNN feature extractor g that extracts hierarchical feature maps F from images I, followed by a classifier h that generates pixel-by-pixel prediction to fit labels S. Together they form a segmentation model f to predict class probabilities for each of the pixels in a given image I:

$$f(I) = h(g(I)). \tag{1}$$

Training is to find an optimal f from the space of functions which minimizes a loss function L that measures the distance between S and $f(I)$:

$$f^* = \operatorname*{argmin}_f L(S, f(I)). \tag{2}$$

The corresponding optimal feature extractor, a.k.a., representations, g^* from $f^*(I) = h^*(g^*(I))$ can be used as the initialization of g for another dataset.

PU Learning Consider a dataset containing N training examples $(x_i, y_i, s_i), i \in \{1, 2, \dots, N\}$, where x_i is the observed features for the i-th examples, and $y_i \in \{-1, +1\}$ is the label for the i-th example, and s_i is a binary variable denoting whether the label for the i-th example is observed or not.

In a normal binary classification setup, labels for all examples are observed:

$$s_i = 1, \forall i \in \{1, 2, \dots, N\} \tag{3}$$

However, in a PU learning setup, only a subset of the positive examples P are observed while the rest are not:

$$s_i = \begin{cases} 1, & y_i = +1 \wedge i \in P \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The set of labeled positive examples is called the positive P set and the other examples form an unlabeled U set.

In this work, we train classifiers by assigning examples in the unlabeled set negative labels and covert PU learning to a learning problem with reliable positive labels and unreliable negative loss. The observed label for the i-th example is:

$$\tilde{y_i} = \begin{cases} +1, & y_i = +1 \wedge i \in P \\ -1, & \text{otherwise.} \end{cases} \tag{5}$$

4 Class-dependent sigmoid loss for PU Learning

In this section, we introduce losses for learning with positive and unlabeled examples.

Class-weighted loss A class-weighted logistic loss $l_{weighted}(\cdot)$ for a pair of input feature and observed label (x, y) with a model $f(\cdot)$ parametrized by θ is:

$$l_{weigted}(x, y; \theta) = \begin{cases} -\alpha \log(\sigma(f(x; \theta))), & y = +1 \\ -\beta \log(1 - \sigma(f(x; \theta))), & y = -1, \end{cases} \tag{6}$$

where α and β are weights for positive and negative class respectively, and $\sigma(\cdot)$ is the sigmoid function. This loss is referred to as the **class-weighted loss** in the rest of paper. Empirically, the choice of α, β can be made based on the

Losses and derivatives

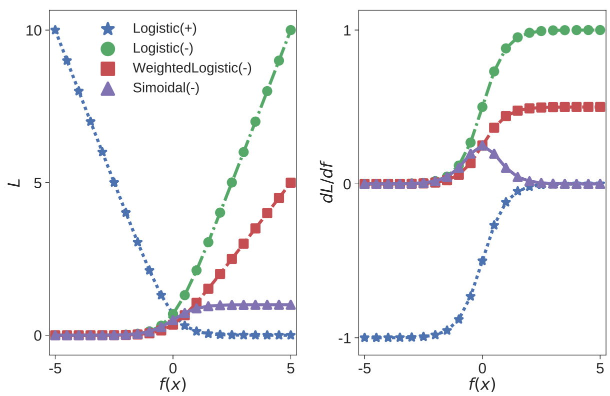

Figure 2: The differences in losses (left figure) and derivatives (right figure) with respect to the model f between the weighted logistic loss and the sigmoid loss for the negative class. The x-axis denotes the model output $f(x)$, varying from negative infinity to positive infinity. We present only $(-5, 5)$ for compact figures. The **+** sign represents the loss of positive samples, and the **-** sign stands for the loss of negative samples. The sigmoid loss of negative examples reaches a plateau and the derivative drops to zero in the region of large $f(x)$, whereas the weighted logistic loss for negative is a linearly scaled logistic loss. The sigmoid loss fulfills the requirement of not punishing the model more for more positive output than less positive output.

highest precision and recall achieved on a validation set, or based on a class priors estimation[5].

We extend the class-weighted logistic loss to a class-weighted cross entropy for multiclass classification with K relevant classes and one one-relevant class (class 0). Suppose there are K relevant categories and one non-relevant categories, the corresponding class-weighted loss l_{wtd} for a training sample (x, y) with a model $f(\cdot)$ parametrized by θ is:

$$l_{weighted}(x, y; \theta) = \begin{cases} -\beta \log(\sigma_0(f(x; \theta))), & y = 0 \\ -\alpha_1 \log(\sigma_1(f(x; \theta))), & y = 1 \\ & \vdots \\ -\alpha_K \log(\sigma_K(f(x; \theta))), & y = K, \end{cases} \quad (7)$$

where $\alpha_1, \ldots, \alpha_K, \beta$ are the weighting factors, $\sigma_0, \ldots, \sigma_K$ are the softmax functions for class 0 to K respectively. This loss is referred to as the **class-weighted cross-entropy**.

Sigmoid/softmax Loss for the negative class The class-dependent sigmoid loss $l_{sigmoid}$ a sigmoid loss for the negative class and keep the loss for the positive class unchanged uses a logistic loss, for example, a logistic loss:

$$l_{sigmoid}(x, y; \theta) = \begin{cases} -\log(\sigma(f(x; \theta))), & y = +1 \\ \sigma(f(x; \theta)), & y = -1, \end{cases} \quad (8)$$

where (x, y) is a pair of input feature and label, and $f(\cdot; \theta)$ is the model parametrized by θ, and $\sigma(\cdot)$ is the sigmoid function. This loss is referred as the **class-dependent sigmoid loss** or the **sigmoid loss** in short, in a sense it uses a sigmoid function for the negative class only.

Figure 2 shows the differences in losses and derivatives with respect to model output between weighted logistic loss and sigmoid loss. The main feature of sigmoid loss for negative examples is its small changes in the region of confident positive, compared to the weighted loss with $\alpha = 1$ and $\beta = 0.5$. As a consequence, the corresponding derivative decreases to zero as the model prediction increases in the positive direction.

The class-dependent sigmoid loss is extendable to a multiclass scenario where K is the number of relevant classes and 0 denotes the non-relevant class. The corresponding loss $l_{softmax}$ for an example, label pair (x, y) with a model $f(;\theta)$ is:

$$l_{softmax}(x, y; \theta) = \begin{cases} 1 - \sigma_0(f(x; \theta))), & y = 0 \\ -\log(\sigma_1(f(x; \theta))), & y = 1 \\ & \vdots \\ -\log(\sigma_K(f(x; \theta))), & y = K, \end{cases} \quad (9)$$

where $\sigma_0, \ldots, \sigma_K$ are the softmax functions for class 0 to K respectively. This loss is called the **class-dependent softmax loss** or the **softmax loss** for simplicity.

Hard bootstrapping loss for the negative class In addition to the proposed sigmoid loss, we also modify the hard bootstrapping loss by Reed et al. [32] for PU learning to set a benchmark. The modified class-dependent hard bootstrapping loss for a pair of inputs and label (x, y) with a model $f(\cdot;\theta)$ is:

$$l_{bootstrap}(x, y; \theta) = \begin{cases} -\log(\sigma_{+1}(f(x;\theta))), & y = +1 \\ -\beta \log(\sigma_{-1}(f(x;\theta))) - (1-\beta)\log(\sigma_{\hat{y}}(f(x;\theta))), & y = -1, \end{cases} \quad (10)$$

where $\hat{y} = \text{argmax}_{j\in\{-1,+1\}} \sigma_j(f(x;\theta))$ is the class with the highest predicted probability and $0 < \beta < 1$ is a hyperparameter to tune. The first term of the loss for the negative class is a weighted logistic loss and the second term can be considered as a regularization term to encourage consistent predictions. This loss is referred as the **bootstrapping loss** for the rest of this paper.

Similarly as the weighted loss and the sigmoid loss, this hard bootstrapping loss $l_{bootstrap}$ can be extended to multiclass:

$$l_{bootstrap}(x, y; \theta) = \begin{cases} -\beta \log(\sigma_0(f(x;\theta))) - (1-\beta)\log(\sigma_{\hat{y}}(f(x;\theta))), & y = 0 \\ -\log(\sigma_1(f(x;\theta))), & y = 1 \\ & \vdots \\ -\log(\sigma_K(f(x;\theta))), & y = K, \end{cases} \quad (11)$$

where $f(\cdot)$ is the model parametrized by θ, and $\sigma_0, \ldots, \sigma_K$ are the softmax functions for class 0 to K respectively, and (x, y) is a pair of example and label, and K is the number of relevant class while 0 is the non-relevant class, and lastly $0 < \beta < 1$ is a hyperparameter.

Implementation details We introduced the sigmoid loss only after training with a class-weighted cross entropy loss for a few epochs. The sigmoid loss of negative examples saturates for very positive outputs, meaning that the confident, positive prediction has little contribution to the weights update. The wrong confident predictions can introduce problems at the beginning of the training procedure when the confident predictions are likely to be made at random. Otimization would reach the plateau when the model made all positive predictions with high confidence. Besides, we also introduce the modified hard bootstrapping loss only after a few epochs trained with class-weighted loss because it also relies on a nonrandom model for sufficiently reliable prediction $\hat{y}$.

Another problem encountered in the PU learning setup is the class imbalance introduced by negatively labeled positive samples. A balanced dataset can become imbalanced in the presence of false negative labels, especially if only a small portion of positive samples are correctly labeled. We reweighed positive and negative samples based on their occurrences of the observed labels to alleviate the influence of imbalance for training. Note that the class-weighted logistic loss reweighed the classes in addition to this frequency balancing class weight.

5 Experiments

5.1 Learning with only positive and unlabeled samples

In this section, we apply the sigmoid/softmax class to the unlabeled examples as if they are negative examples and train classifers with only positive data and unlabeled data.

5.1.1 2D non-linear dataset

To investigate the decision boundaries led by the sigmoid loss for the negative class, we trained a two-layer multilayer perceptron, with six neurons per layer, using the normal logistic loss, the class-weighted logistic loss, and the class-dependent sigmoid loss respectively and visualize the optimal decision boundaries.

Experimental setup The training data contains four hundred samples per class drawn randomly from two interleaving half circles with noises added with a minor standard deviation, as shown in Figure 3. Half of the positive examples were assigned negative labels, resulting in a training data with reliable positive labels but noisy negative labels. The three different losses were trained with this noisy training data, and the result decision boundaries are drawn as white regions in Figure 3. The same multilayer perceptron classifier was also trained with true labels to present a baseline decision boundary. The weights for positive class and negative class in the weighted logistic loss were chose to be 1 and 0.5 respectively.

Results If trained with the sigmoid loss, the decision boundary is distant from the positive cluster with a relatively large margin as shown in Figure 3. By contrast, the weighted logistic loss results in a decision boundary still closed to the positive examples. For sigmoid loss, the mislabeled positive examples far away from the decision boundary do not contribute more loss than samples less distant from the decision boundary. As a consequence, the loss derivative with respect to the model weights is larger for uncertain predictions in overall than for confident predictions, illustrated by marker sizes in Figure 4. Therefore, training with the sigmoid loss emphasizes the positive predictions with low confidence instead of equally down-weighting all incorrect positive predictions. The emphasization of uncertain predictions by sigmoid loss leads to a margin from the decision boundary to both positive examples and negative examples.

5.1.2 CIFAR dataset

To compare the precision and recall achieved by the class-dependent softmax loss and the class-weighted cross entropy, we trained a CNN classifier to distinguish images of multiple relevant categories from non-relevant images when relevant images are partially labeled.

Experimental setup We trained an eight-layer CNN model to classify images into eleven classes: ten relevant classes from CIFAR10 and one non-relevant class for all categories from CIFAR100. Relevant images are partially labeled, and the rest forms an unlabeled (U) set together with the non-relevant images. An eight layer CNN model was trained with the cross-entropy loss, the class-weighted cross-entropy loss and the class-depend sigmoid loss respectively in the simulated PU learning setup, where 50% of the positive examples were unlabeled. The CNN model was also trained with the modified hard bootstrapping loss introduced in Section 4 to set a benchmark for the state-of-the-art method to learn in the presence of label noises. Model performances were evaluated on a separate test set with true labels. The architecture of the CNN model can be found in Appendix C. Each model was trained from scratch with Adam optimizer and base learning rate 0.0001. Experiments were repeated three times with random split of P set, and U set and standard deviations were around 0.01 if not explicitly mentioned.

Results Table 1 shows using the softmax loss for the non-relevant class achieves better recall than the class-weighted cross-entropy loss without lowering precision significantly. With 50% of the relevant examples correctly labeled and the rest assigned non-relevant labels, the normal cross-entropy loss leads to an imbalanced model with high precision but low recall, and therefore with a low f1-score. By reweighing the loss for the non-relevant class by a factor of 0.5, the model becomes balanced for precision and recall so that the result f1-score is improved significantly. Compared to the class-weighted cross entropy, the class-dependent softmax loss improves recall by 0.08 while reduces precision only by 0.01. The f1-score achieved by the class-dependent softmax loss is slightly better than the class-weighted loss, though not as good as training with clean labels with either 50% of the sample or the complete training set. The state-of-the-art benchmark method, the hard bootstrap loss, achieves the same f1-score as the softmax loss but not as high recall. This result is as expected because the softmax loss and the hard bootstrap loss share an idea of encouraging confident predictions.

By varying the labeled percentage of the relevant images, Figure 5 demonstrates that the class-dependent softmax loss performs slightly better than the class weighted cross-entropy when the labeled percentage of relevant images is neither too high (> 0.8) nor too low (< 0.2).

5.2 Learning with incomplete segmentations

To compare the class dependent sigmoid loss with the normal logistic loss, and class-weighted logistic loss for training foreground/background segmentation with incomplete segmentations, we used again the PASCAL VOC2011 dataset with extra segmentation [13].

Dataset PASCAL VOC2011

Experimental setup We simulated inexhaustive segmentations the same way as described in Section 5.3. The same AlexNet-FCN model was trained together with the different loss functions to predict binary segmentation, determining whether a pixel belongs to an object or not. The same hyperparameters for optimization were used as in Section 5.3. The trained models were evaluated with the test set of PASCAL VOC2011 segmentation dataset with binary segmentations.

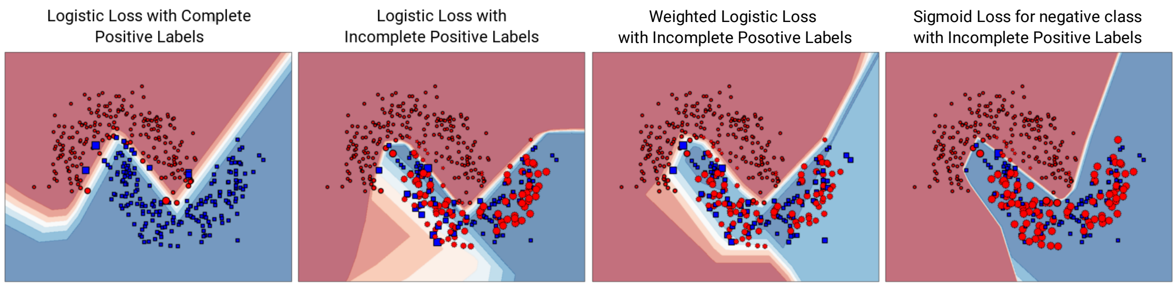

Figure 3: Decision boundaries of a 2-layer multilayer perceptron trained with different losses on a 2D moons dataset with the unlabeled positive. A **red circle** indicates an example labeled as positive and a **blue square** indicates the example has a negative label. The **background colors** represent the classifier prediction in the corresponding area: **red** for negative class, **blue** for positive class and **white** for the class transition areas, i.e., decision boundaries. The **markers sizes** demonstrates the training loss normalized per-class. Compared to the normal logistic loss and weighted logistic loss (positive:negative=1:0.5), the decision boundary optimized with the sigmoid loss has a larger margin from the positive and negative clusters. (Best viewed in color)

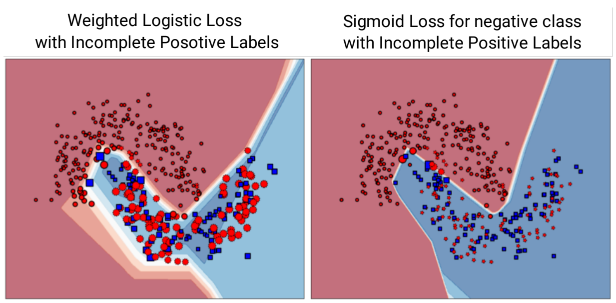

Figure 4: Derivatives w.r.t the last layer output for the two losses (normalized per class and shown as the marker size). The sigmoid loss has small derivatives for predictions farther from the decision boundary. (Best viewed in color)

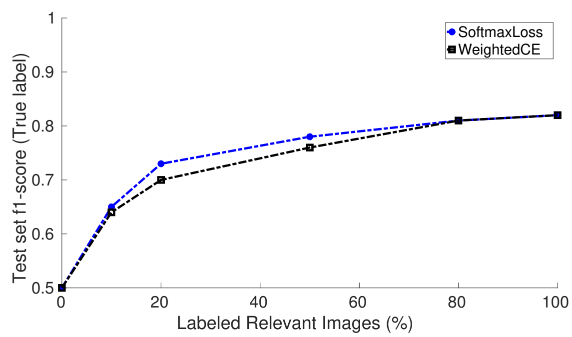

Figure 5: Comparing f1-score for the class-dependent softmax loss and the class-weighted cross entropy with varying percentage of relevant images labeled. The class-dependent softmax loss achieves better test f1-score than the class-weighted cross entropy when 20% and 50% percentage of relevant images are labeled, and the others are mixed with non-relevant images.

Results As shown in Table 2, the class dependent sigmoid loss achieves the highest mean accuracy, approximately 0.07 better than training with the normal logistic loss, and 0.04 better than the class-weighted loss. In contrast to the improvement of mean accuracy, improvement of mean IU for both the class-weigthed loss and the class-dependent logistic loss are insignificant. The increase in the mean accuracy is caused by an increase in foreground accuracy and a decrease in background accuracy. The decrease in background accuracy interferes the improvement mean IU since the mean IU counts for both low false positive rate and low false negative rate.

Selective predictions made by the models trained with the sigmoid negative loss and the cross entropy loss were presented in Figure 6. For the two example images shown,

Annotation	Loss	acc.	mean prec.	mean rec.	mean F_1
R+N	CrossEntropy	0.87	0.88	0.82	0.85
50%(R+N)	CrossEntropy	0.83	0.84	0.78	0.80
50%R+U	CrossEntropy	0.66	0.94	0.38	0.49
50%R+U	ClassWeighted	0.78	0.75	0.75	0.76
50%R+U	SoftmaxLoss	0.79	0.74	**0.83**	**0.78**
50%R+U	BootstrapHard	**0.80**	0.76	0.81	**0.78**

Table 1: Comparing different losses for training a 2-layer multilayer perception to classify ten relevant classes and one non-relevant class with partially labeled relevant examples and unlabeled non-relevant examples. The trained classifiers are evaluated on a test set of true labels. For each of the relevant classes, precision, recall, and f1-score are measured with the one-vs-all strategy and averaged. **R+N** denotes model trained with the complete relevant labels (R set) and non-relevant labels (N set); **50%(R+N)** represents model trained with the half of the relevant labels and non-relevant labels respectively; **50%R+U** means the model is trained with half of the relevant samples, and the rest relevant samples are mixed with non-relevant samples (U set). Weighting the non-relevant class by a factor of 0.5 improves the mean f1-score from 0.49 to 0.76. Both the class-dependent softmax loss and the hard bootstrapping loss perform even better than simply weighing the classes, but not as good ass training with a set of labeled negative examples. Using the softmax loss for non-relevant class achieves the highest mean recall, whereas the modified hard bootstrapping loss has a higher accuracy with 50% positive examples unlabeled.

Annotation	Loss	overall acc.	mean acc.	f.w. IU	mean IU
Complete	LogisticLoss	0.90	0.85	0.82	0.75
50%Unseg.	LogisticLoss	0.85	0.68	0.73	0.60
50%Unseg.	ClassWeighted	0.84	0.71	0.73	**0.62**
50%Unseg.	SigmoidLoss	0.83	**0.75**	0.72	**0.62**

Table 2: Training foreground/background segmentation with different losses when 50% of the objects are unsegmented. The performances are achieved on the test set of PASCAL VOC2011 segmentation dataset. Both the class-dependent sigmoid loss and the class-weighted logistic loss perform better than the normal logistic loss when 50% objects unsegmented but not as good as the model trained with complete segmentations. The class-dependent sigmoid loss has a better mean accuracy than the class-weighted logistic loss and a similar mean IU as the class-weighted logistic loss.

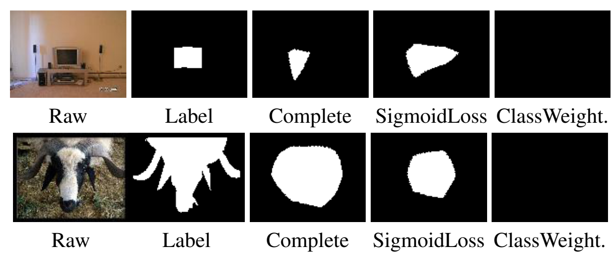

Figure 6: Example predictions made by models trained with the logistic loss and the class-dependent sigmoid loss. This figure presented two selective images for which the model trained with the logistic loss failed to segment objects, whereas the model trained with the class-dependent sigmoid negative loss succeed.

the model trained with the cross entropy loss failed to segment objects from images whereas sigmoid negative loss predicted segmentations on the position of the objects. The coarse outlines were mainly due to the limited compacity of the FCN-AlexNet model. The third column shows predictions given by model trained with complete training segmentation, and it did not produce more accurate outlines. There is no example observed correctly segmented by the model with the class-weighted logistic loss but not by the model with the class-dependent sigmoid loss.

5.3 The influence of label noises in segmention

To investigate the influence of the lablel noises, (1) inexhaustive segmentations, (2) objects mislabelling and (3) false positive segmentations, on the learned representations, we set up experiments with simulated label noises from a well-annotated dataset, the PASCAL VOC2011 segmentation dataset [10].

5.4 Label noises simulation

It is challenging to find a dataset with both clean and noisy labels available, so we simulate segmentation noises with correct labels. A straightforward way to simulate noisy labels is to corrupt true labels stochastically for each segment with a corruption model. The corruption model simply describes the probability of the observed labels given the true labels.

For segmentation problems, each pixel (or voxel for 3D segmentation) of a training image has a label assigned to one of the pre-defined categories. Supposing there are K pre-defined categories, the label of a pixel in the i-th row and j-th column from an image of size $h \times w$:

$$y_{ij} = \begin{cases} 1 \leq k \leq K, & \text{for foreground pixels} \\ 0, & \text{for background pixels} \end{cases}$$

where $1 \leq i \leq h, 1 \leq j \leq w$ and $i, j, k \in \mathrm{Z}^+$.

Inexhaustive segmentation Given the true labels, which pixels belong to the same object is known. Inexhaustive segmentation is simulated by flipping all pixels for an object of the k-th category from k to 0 with probability p_{0k}. The probability for these pixels to have correct labels is then: $1 - p_{0k}$.

Objects Mislabelling To simulate objects mislabelling, we stochastically convert pixel labels of segment for an object of the k-th category from k to j with probability p_{jk}, where $j, k \leq K$ and $j, k \in \mathrm{Z}^+$. Probabilities for all possible combinations of j and k form a *confusion matrix* in which probabilities in each column sum up to 1.

False positive segmentations The presence of false positive segmentations is in two steps: excluding classes except one from the foreground categories, forming the clean labels set, and assigning pixels of the excluded categories with foreground labels with a probability of p_{11}, constructing the noisy labels set.

Dataset In this experiment, fifteen out of twenty categories of the VOC2011 dataset were selected to form a *pre-training dataset* and the other categories formed a *fine-tuning dataset*. The pre-training dataset was used to train a Fully Convolutional Network with AlexNet (FCN-AlexNet) model [26] for segmentation in the presence or absence of simulated label noises. Inexhaustive segmentations, objects mislabelling, and false segmentations were simulated independently with stochastical corruptions to the well-annotated pre-training dataset, followed the descriptions in Section **??**. The fine-tuning dataset was used to fine-tune the weights of convolutional layers from the pre-trained FCN-AlexNet models. To avoid that the choice of pre-training and fine-tuning splitting for categories influence the results, the 20 categories of VOC2011 were divided equally into four folds. The training dataset was enriched with extra segmentations by Hariharan et al. [13] To keep the segmentation task simple, we used only single-object images, resulting in totally 4000 training images for 20 categories available for pre-training, fine-tuning and evaluation. We subsampled the original images by four times to accelerate the training process.

Experimental setup The fine-tuned models were evaluated by mean intersection over union ratio (mean IU) achieved on the fine-tuning test set, referring to as the *fine-tuning performance*. Performance improvement of fine-tuning transferred models compared to a randomly initialized model indicates the transferability of pre-trained weights. The non-transferable layers of FCN-AlexNet were randomly initialized with Xavier Initialization. Experiments run for each fold independently, and the exact partitions of each fold are listed in Table 3. Fully Convolutional Networks with AlexNet was used for experiments because of its relatively small capacity and thus short training time. The pre-trained AlexNet model was used to set a baseline of performance, denoted as the ImageNet model. The ImageNet model and completely random weight initialization were considered as the upper baseline and lower baseline, respectively, for various pre-trained weights summarized in Table 3. The default hyperparameters of FCN-AlexNet in [26] were kept unchanged. The training process run 240,000 iterations for pre-training phase, and 12,000 iterations for fine-tuning phase. Snapshots for trained models were taken every 4,000 iterations. Each experiment was repeated three times, mean and standard deviation were computed over the last five snapshots for all repetitions.

Objects mislabelling The negative influence of random object labels on feature transferability is demonstrated in Table 3. Compared to the model trained with true labels, both models trained with all random labels and half true half random labels are less transferable to the fine-tuning dataset. Transferring the AllRandomLabels model or the HalfRandomLabels model is no better than randomly initializing model weights. The mislabeled objects in segmentations negatively impact the transferability of CNN models in this simulated experiment setup.

We then binarize pre-training classes to foreground and background and achieve a fine-tuning performance better than training with the exact (half) randomized labels and equivalent to training with correct labels. Randomized object labels were mislabeled among foreground classes so that binarizing labels a foreground and background classes can in a sense correct the randomized labels. Compared to the precise but inaccurate noisy labels, binarized labels are accurate but imprecise. This observation indicates that inaccurate labels have a larger influence on the learned representations than imprecise labels.

Precise labels vs. Accurate labels To validate that accurate, imprecise labels have less negative effect on feature transferability than inaccurate, precise labels. In figure 7, we decrease the preciseness of pre-training labels by grouping the pre-training classes into a various number of categories to demonstrate the corresponding change of feature transferability. Addition to the binary categorization, we also categorized the fifteen pre-training classes into four meaningful categories: person, animal, vehicle, indoor according to [10]. The result transferred and fine-tuned model is shown as the error bars on the solid line at categories=4. It has almost the same fine-tuning performance as the model trained with binarized labels and the model trained with precision labels (shown as error bar at categories=15).

As a comparison, the fifteen classes were randomly categorized into 4, 7, 11 categories and shown as isolated error bars in figure 7 at categories=4, 7, 11 respectively. Figure 7 reveals that reducing label preciseness by categorizing pre-training classes has little effect on the fine-tuning performance of transferred models. Even categorizing classes at random without explicit meaning can pre-train weights better than random initialization (shown as error bar at categories=0).

In contrast to the significant influence of randomized precise labels, the imprecise binarized labels have little effect to the learned representations. Binarizing is helpful to learn better representations when mislabeled objects is dominant.

Inexaustive segmentation Fine-tuning performance for pre-trained models with complete segmentations and incomplete segmentations are summarized in Table 3. Pre-training data with half of the objects unsegmented results in fine-tuned models with a worse mean IU (average across four folds) than pre-training with complete labels by 0.04, the same as no pre-training. This observation demonstrates that inexhaustive segmentations have a negative impact on representations transferability.

By applying the class-dependent sigmoid loss to pre-train models with half of the objects segmented, a fine-tuning performance comparable to the model pre-trained with complete segmentation is achieved. The class-dependent sigmoid loss compensates the negative influence of inexhaustive segmentations on the fine-tuning performance of the pre-trained model. Note that we train foreground/background segmentation instead of multi-class segmentation when applied the class-dependent sigmoid loss because binary segmentation can produce as good representation as multi-class segmentation.

False positive segmentations We observe a little influence of the incorrectly segmented objects from the non-predefined but meaningful categories, i.e., the false positive segmentations, on the learned representations. To simulate the false positive segmentations errors, we selected one category, either cat or dog depending on the folds, as the target category and all the other 14 categories in the pre-training dataset became non-target, as discussed in Section **??**. In the presence of false segmentations, instances from non-target categories can be misannotated as the target category with a probability of $p_{10} = 0.5$. The result pre-trained models is named as the HalfFalsePositive model in Table 3. The noise-free counterpart of is the model pre-trained with segmentations of the selected target category only, and the other 14 categories remained unsegmented, denoted as the NoFalsePositive model.

We observe from Table 3 that transferring the HalfFalsePositive model performs almost the same as the NoFalsePositive model. Therefore, false positive segmentation for objects from the 14 categories have a little nega-

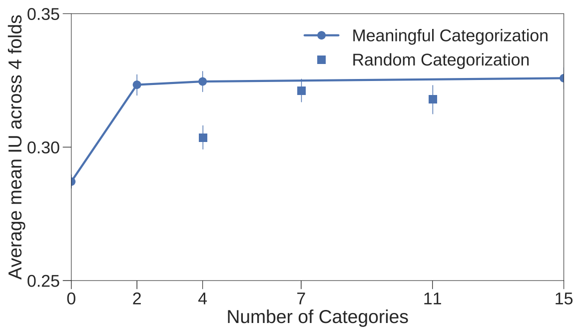

Figure 7: Test performance for the fine-tuned models pre-trained with varying categorization of pre-training classes. Zero categorize means no pre-trained weights used and the model was random initialized. Error bars located on lines denote the meaningful categorization, and isolated error bars denote random categorizations (RC) of the 15 classes. The displayed mean IU/mean accuracies and standard deviations were averaged over four folds listed in Table 3. The line shows that binarizing and categorizing classes meaningfully had little negative effect on feature transferability.

tive impact on pre-training CNN models.

6 Discussion

We investigate in this work the influence of label noises but not segmentation noises for a noisy segmentation dataset. In practice, noisy segmentation dataset may also contain segmentation errors such as imprecise boundaries, over-segmenting and under segmenting the objects. Future studies are necessary to learn representations in the presence of segmentation noises.

When we simulate the label noises in our experiments, we assume unsegmented objects and mislabeled objects occur independently to the shape and appearance of objects. In practice, some categories may have a higher probability to be mislabeled due to its ambiguity in shapes or appearances, such as bear and teddy bear. Similarly, ambiguous objects are more likely to be missing than easily recognizable objects. Experiments on a real dataset with both clean and noisy segmentation are valuable to validate that our findings.

We proposed to learn representations with binary segmentation when objects mislabelling dominates the pre-training dataset. Training binary segmentations can also be relevant when there are multiple segmentation datasets for pre-training, and category overlappings and affinities prevent from combining multiple datasets into one.

We argued to not over-punish confident, positive predictions when negative labels are noisy so that we come up with the sigmoid loss for negative examples. But with the sigmoid loss, we are not able to determine what is the threshold of confident predictions. A generalized logistic function may be used to replace the normal logistic function as the activation function to achieve a more flexible S-shape and the tuning where the loss saturates. For example, a parametrized sigmoid loss for the negative class could be $l_- = \alpha(\frac{1}{1+\exp{(\beta z)}})^\gamma$, where z is the model output for the negative class, α is the scale factor, γ affects where the loss starts, β determines where the loss saturates.

Besides, saturating loss for confident predictions has an effect of encouraging confident predictions. It is intuitively similar to the minimum entropy regularization for semi-supervised learning, which also encourages confident predictions. Both the class-dependent sigmoid loss for PU learning and the minimum regularization scheme favors low-density separations of input features. The hard bootstrapping loss by [32] has a “soft” alternative which is equivalent to the minimum entropy regularization. This similar in the sigmoid loss and the bootstrapping loss explains why the sigmoid loss has a similar performance as the hard bootstrapping loss.

Not over-punishing confident predictions have also it disadvantages: (1) If a classifier makes incorrect predictions with high confidence, it tends to keep being wrong for these examples and emphasize predictions by itself. (2) Punishing confident predictions more than uncertain predictions with the logistic loss is a design of choice for neural networks to optimize more effectively, whereas the sigmoid loss breaks it. These factors determine that the sigmoid loss often performs worse than the logistic loss when the dataset contains only correct labels or a few noisy labels. There is a trade-off to make between punishing and not punishing more for confident predictions, based on the prior knowledge: an estimation of the noisy

Fine-tuning categories	Initial Feature Extractor	Fine-tuning mean IU per pretraining-finetuning fold				Average mean IU across four folds
		aeroplane, bicycle, bird, boat, bottle	bus, car, cat, chair, cow	dining table, dog, horse, motorbike, person	potted plant, sheep, sofa, train, TV	
Baseline	ImageNetModel	0.42 ± 0.01	0.51 ± 0.01	0.49 ± 0.01	0.47 ± 0.01	0.47 ± 0.01
	RandomWeights	0.29 ± 0.01	0.29 ± 0.03	0.27 ± 0.01	0.30 ± 0.02	0.29 ± 0.02
Objects Mislabelling	TrueLabels	0.29 ± 0.01	0.36 ± 0.01	0.29 ± 0.01	0.37 ± 0.01	$\mathbf{0.33 \pm 0.01}$
	AllRandomLabels	0.29 ± 0.01	0.33 ± 0.03	0.26 ± 0.01	0.28 ± 0.01	0.29 ± 0.01
	HalfRandomLabels	0.27 ± 0.01	0.33 ± 0.02	0.25 ± 0.01	0.29 ± 0.01	0.29 ± 0.01
	BinarizedLabels	0.30 ± 0.02	0.35 ± 0.01	0.29 ± 0.02	0.35 ± 0.03	$\mathbf{0.32 \pm 0.02}$
Inexaustive segmention	CompleteLabels	0.29 ± 0.01	0.36 ± 0.01	0.29 ± 0.01	0.37 ± 0.01	$\mathbf{0.33 \pm 0.01}$
	HalfUnsegmented	0.26 ± 0.01	0.30 ± 0.03	0.28 ± 0.03	0.32 ± 0.02	0.29 ± 0.02
	SigmoidLoss	0.30 ± 0.01	0.37 ± 0.01	0.31 ± 0.02	0.34 ± 0.02	$\mathbf{0.33 \pm 0.02}$
False positive segmentaion	NoFalsePositive	0.26 ± 0.01	0.37 ± 0.03	0.27 ± 0.01	0.33 ± 0.04	$\mathbf{0.31 \pm 0.02}$
	HalfFalsePositive	0.27 ± 0.01	0.34 ± 0.01	0.30 ± 0.01	0.32 ± 0.01	$\mathbf{0.31 \pm 0.01}$

Table 3: Segmentation performance for fine-tuned FCN-AlexNet models pre-trained on 15 categories from the PASCAL VOC2011 dataset and fine-tuned on the other 5 categories. **ImageNetModel** represents the pre-trained ImageNet model; **RandomWeights** indicates that the randomly initialized weights; All the other extractors were pre-trained in the presence/absence of the corresponding label noises listed in the leftmost column. Half of the objects unsegmented (**HalfUnsegmented**) result in pre-trained models not better than random weight initialization. Introducing the sigmoid negative loss in the pre-training phase was able to improve the fine-tuning performance to be comparable to pre-trained model with complete segmentation (**CompleteLabels**). Using random labels (**RamdomLabels**) decreased the fine-tuning performance of transferred models, compared to using true labels (**TrueLabels**). Binarizing the pre-training classes as foreground and background help overcome the negative effects of random labels. Applying the sigmoid loss to the negative class when pre-trained with inexhaustive segmentations achieves a comparable fine-tuning performance to pre-training with the complete segmentations. Including the false positive segmentations in pre-training (**HalfFalsePositive**) achieves the same fine-tuning performance as not including the false positive segmentations (**NoFalsePositive**), better than random initialization.

negative labels percentage.

Lastly, applying the sigmoid loss to segmentation data with incomplete segmentations assumes that the pixels for objects are unlabeled independently. In practice, there is often a spatial dependence for noises in pixel labels which can be valuable to improve performance. For example, if one or a few pixels have a high probability of being foreground, their neighboring pixels are probably also foreground.

Extending PU learning from classification to segmentation In Section 1, we argue that learning with unlabeled foreground pixels is similar to a PU learning setup. Howvever, there is still differences between learning with unlabeled foreground pixels and learning with positive and unlabeled examples.

The first difference is that each example in the normal PU learning setup is independent of each other, whereas pixels in images are not. Assuming the probability of mislabeling foreground pixel as the background is independent of its neighbor pixels, the classification losses can be applied to segmentation problems by performing per-pixel classification problems.

Another difference between incomplete segmentation and a normal positive and unlabeled learning problem is that pixels for objects of various categories can be unlabeled. Supposing there are K categories of interest, varying from class 1 to class K, the class 0 is for unlabeled data which may or may not belong to the K defined categories. The sigmoid loss can be extended train deep learning models with unlabeled examples from various categories:

7 Conclusion

We investigate in this paper to learn representation by training with segmentation datasets containing label noises. Specifically, we report both objects mislabelling and inexhaustive segmentations negatively influence the transferability of learned representation. By contrast, false positive segmentations do not reduce the fine-tuned performance of learned representation as the other two types noises do. We present that binarizing classes as foreground and background slow the decrease of the fine-tuning performance of the learned representations due to the mislabeled objects. Incomplete segmentation causes the trained model to make predictions with a low recall. To overcome the influence of incomplete segmentations, we propose a class-dependent sigmoid loss to not over-punish the confident, positive predictions for samples assigned negative labels. Compared to simply reweighing classes differently, the proposed sigmoid loss for the negative class achieves higher recall while not sacrificing precision by much. Applying the sigmoid loss to the segmentation model pre-training improves both the pre-training and the fine-tuning performance for a pre-training dataset with simulated incomplete segmentations.

References

[1] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36, 2012.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[3] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *International Conference on Machine Learning*, pages 1386–1394, 2015.

[4] Marthinus C du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Advances in neural information processing systems*, pages 703–711, 2014.

[5] Marthinus Christoffel Du Plessis and Masashi Sugiyama. Class prior estimation from positive and unlabeled data. *IEICE TRANSACTIONS on Information and Systems*, 97(5):1358–1362, 2014.

[6] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[7] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM, 2008.

[8] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[9] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Artificial Intelligence and Statistics*, pages 153–160, 2009.

[10] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[13] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 991–998. IEEE, 2011.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[17] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[18] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. Pixel objectness. *arXiv preprint arXiv:1701.05349*, 2017.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[22] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pages 448–455, 2003.

[23] Xiao-Li Li and Bing Liu. Learning from positive and unlabeled examples with different data distributions. *Machine Learning: ECML 2005*, pages 218–229, 2005.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[25] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Building text classifiers using positive and unlabeled examples. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 179–186. IEEE, 2003.

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[27] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59, 2011.

[28] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 567–574, 2012.

[29] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898, 2014.

[30] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[31] Giorgio Patrini, Alessandro Rozza, Aditya Menon, Richard Nock, and Lizhen Qu. Making neural networks robust to label noise: a loss correction approach. *arXiv preprint arXiv:1609.03683*, 2016.

[32] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[33] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[35] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[39] David MJ Tax and Feng Wang. Class-dependent, nonconvex losses to optimize precision. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 3314–3319. IEEE, 2016.

[40] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[41] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.

[42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[43] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

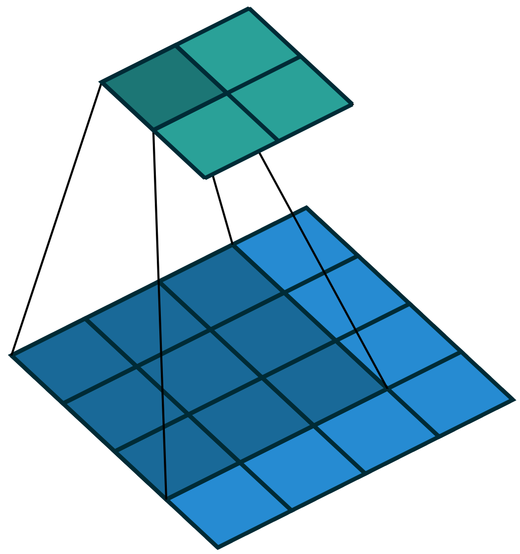

Figure 8: A basic convolution operation. A 3x3 convolutional filter convolves with a 3x3 window sliding over the image (bottom). The output of convolutions at each sliding position form a feature map (top). This figure was drawn by Dumoulin and Visin [6]

A Convolutional Networks for Semantic Segmentation

A.1 Convolutional Neural Networks

The main components of a typical convolutional neural network (CNN) are several layers of convolutions and sub-sampling, followed by a few fully-connected layers.

An example CNN model, LeNet-5 (1998) [20], is shown in Figure 9. The first convolutional layer of LeNet contains 6 convolutional kernels of size 5x5 and each convolutional kernels convolve with small windows sliding over the images and produce a feature map of size 28x28. Each output in the produced feature map is corresponding to a small sub-region of the visual field (the image), called a *receptive field*. A following max pooling layer subsamples the feature maps by a factor of two by extracting the maximum values for every two adjacent pixels literally

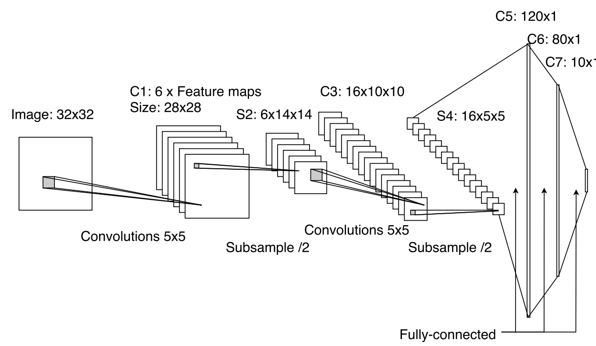

Figure 9: An example convolutional neural network, LeNet-5 [20]

and vertically. The result feature map S2 has a shape of 14 by 14 and a receptive field of 6 by 6. Another sequence of convolutional and pooling layers generate feature maps of size 5x5 with receptive field 16x16. Neurons in the last three layers of LeNet are fully connected to the layer before and the layer after if exists, creating the final prediction for 10 classes.

Features produced by CNN models have a rich hierarchy varying from local to global, from simple to complex. The bottom layers in the convolutional layer stack have smaller receptive fields and the top layers have larger receptive fields. A small receptive field means that the filter have access to information only in a local sub-region of the image while a large receptive field can convey more global information.

The various pattern responses from local to global, from simple to complex for stacked convolutional layers is a reflect of emulating animals visual cortex. In cat's visual cortex [17], two basic cell types of visual cortex have been identified: Simple cells respond maximally to specific edge-like patterns within their receptive field. Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern. The shallower convolutional layers play a similar functionlity as simple cells while the deeper layers maps are similar to complex cells.

The main benefit of CNN compared to a standard multilayer neural network (multilayer perceptron) is that 1. take advantage of the 2D structure of an input image 2. it is easier to optimize because of spatial weights sharing and local connectivy pattern of convolutional layers. Convolutional neurons and maximum pooling, translation invariance as well as scaling invariance and distortion invariance to some extent are achievable for convolutional neural networks. [20] Different from the traditional handcrafted features, learnable convolutional features normally generalize well and can achieve better performance for dataset with a complex input distribution. [19] By increasing the number of convolution layers and number of filters in each layer, one can create CNN models with high capacity, meaning a large space of representable functions. This can be beneficial for datasets of immense complexity, for example, ILSVRC [34], Microsoft COCO [24], as long as there are sufficient training samples with an appropriate optimization strategy.

A.2 Semantic image segmentation

Semantic image segmentation is to segment images into semantically meaningful partitions, a.k.a.,*segments*. It can be operated as classifying pixels into the corresponding pre-defined categories.

CNN models on object classification tasks can be adapted to perform semantic image segmentation tasks. [26] One of the primary challenges of applying CNN model to segmentation tasks is how to combine global information and local information to solve semantics and localization altogether. In contrast to object classification tasks, which normally only need global information to resolve semantics, segmentation tasks also require local information to resolve locations.

Long et al. [26] proposed a so-called skip architecture in the Fully convovolutional networks (FCN) to aggregate information from the local low-level features in the hierarchy with global information from the high-level features. As we discussed in the previous session, convolutional layers can extract hierarchical features, varying from low-level to high-level encode information from local to global. The low-level features are fine, presenting appearances and the high-level features are coarse, revealing semantics. By combining them together, it becomes possible to create accurate and detailed segmentation.

Convolutional layers in FCN for feature extractions (solid arrows in Figure 10) can be transferred from ImageNet models.

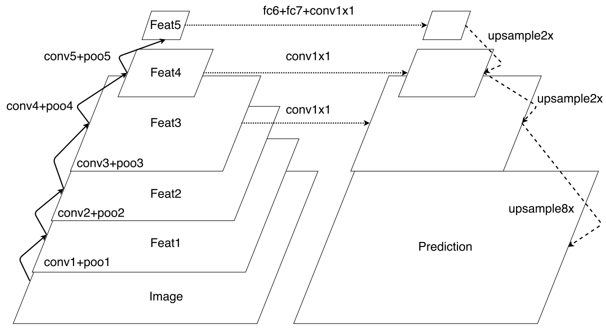

Figure 10: Fully convolutional network (FCN) by Long et al. (2015) [26]. Features of different resolutions are stacked in a feature pyramid on the left-hand side, with the image at the bottom of the pyramid. Predictions of different resolutions are piled in a prediction pyramid on the right-hand side. Each solid arrow denotes a few convolutional layers followed by a max polling layer; Dotted arrow represents convolutional layers with kernel size one by one; Dashed arrows are up-sampling layers or transposed convolutional layers. From top to bottom, each level of prediction is upsampled and merged with the prediction under it. The bottom prediction is output as the final prediction, which has the same size as the image.

B Cost function and Optimization

B.1 Cross entropy loss

The cross entropy loss (a.k.a. softmax loss) is one of the most commonly used cost function for convolutional neural networks in classification problems. Let $x^{(i)}$ be an input example from totally m examples, $y^{(i)} \in 0, ..., K$ be the corresponding label, and θ be parameteres of model $f(\cdot)$. The cross entropy loss is defined as:

$$J(\theta) = -\sum_{i=1}^{m}\sum_{k=0}^{K} 1\{y^{(i)} = k\} \log P(y^{(i)} = k | x^{(i)}; \theta)$$

In the equation above, $1\{\cdot\}$ is the "indicator function" defined as:

$$1\{\text{statement}\} = \begin{cases} 1, & \text{statement is true} \\ 0, & \text{otherwise} \end{cases}$$

$P(y^{(i)} = k|x^{(i)}; \theta) = \sigma(f(x;\theta))_k$ is the likelihood of $y^{(i)}$ being k, predicted by model $f(\cdot)$, where $\sigma(\cdot)_k$ is the softmax function that applies to model output for the k-th class.

Model outputs $f(x^{(i)}; \theta)$ is a vector of k elments with values varying from negative infinity to positive infinity. Each element of the output vector is corresponding to one class out of K classes. A larger output value for one class, k, than another, j, means that the example $x^{(i)}$ is more likely to be class k than class j. The softmax function ensures that the model outputs are normalized to a region between 0 and 1, and sum up to 1 for all classes so that the result outputs fullfils a probability distribution over K different possible outcomes.

The cross entropy loss is a form of negative log-likelihood. The loss is closed to zero if the predicted probability of $y^{(i)}$ is large, and takes a large positive value if the probability is small. Minimizing the negative log likelihood of the correct class can be interpreted as performing Maximum Likelihood Estimation (MLE), a commonly optimization.

B.2 Gradient based optimization and Backpropagation

The model is optimized by solving the optimal θ that minimizes the loss function. It is impossible to solve θ for a non-linear model analytically so that a gradient-based optimization can be used as an efficient alternative.

The derivative of the cross entropy loss with respect to the k-th parameter of the last layer $\theta_k^{(L)}$ is:

$$\nabla_{\theta_k^{(L)}} J(\theta) = -\sum_{i=1}^{m} \left[z^{(i)} \left(1\{y^{(i)} = k\} - P(y^{(i)} = k|x^{(i)}; \theta) \right) \right]$$

where the superscription (L) of θ denotes the layer number of the last layer, and $z^{(i)}$ is the output of the last layer for the i-th example.

Weights of the last layer in the $t+1$-th iteration is updated by:

$$\theta_{t+1}^{(L)} = \theta_t^{(L)} - \alpha \nabla_{(\theta^{(L)})} J(\theta)$$

where α is the learning rate determining how quickly the weights are updated.

Gradients in the layers $l < L$ are calculated via a so-called back propagation of errors. The error of l-th layer is propagated the layer after $l+1$:

$$\delta^{(l)} = \left((\theta^{(l)})^T \delta^{(l+1)} \right) \bullet f'(z^{(l)})$$

where $f'(z^{(l)})$ is the derivative of the activation function. Gradients with respect to weights for the l-th layer is:

$$\nabla_{\theta^{(l)}} J(\theta) = \delta^{(l+1)} (a^{(l)})^T$$

The weights update for the l-th layer in the $t+1$ is computed similarly as the last layer by:

$$\theta_{t+1}^{(l)} = \theta_t^{(l)} - \alpha \nabla_{(\theta^{(l)})} J(\theta)$$

layer name	output size	8-layer
conv1	16×16	3×3, 32, LeakyReLU(0.2)
		3×3, 32, LeakyReLU(0.2)
		2×2 max pool, dropout(0.2)
conv2	8×8	3×3, 64, LeakyReLU(0.2)
		3×3, 64, LeakyReLU(0.2)
		2×2 max pool, dropout(0.2)
conv3	4×4	3×3, 128, LeakyReLU(0.2)
		3×3, 128, LeakyReLU(0.2)
		2×2 max pool, dropout(0.2)
fc	1×1	flatten, 512-d fc, ReLU, dropout(0.5)
		11-d fc, softmax
Parameters		1,341,739

Table 4: 8-layer Convolutional Neural Networks used in the classification of the CIFAR dataset.

C Additional information

C.1 An 8-layer Convolutional neural network

The architecture of the 8-layer convolutional neural network used in the classification of CIFAR dataset is presented in Table 4.

C.2 Evaluation metrics

$$\text{accuracy} = \frac{\text{true pos. + true neg.}}{\text{true pos. + false pos. + true neg. + false neg.}}$$

$$\text{precision} = \frac{\text{true pos.}}{\text{true pos. + false pos.}}$$

$$\text{recall} = \frac{\text{true pos.}}{\text{true pos. + false neg.}}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{IU} = \frac{\text{true pos.}}{\text{true pos. + false pos. + false neg.}}$$