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 obtains better segmentation performance as well as better representations 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 [27, 2, 14] to transfer convolutional neural network (CNN) models [20, 37, 39, 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 [25], 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[35], 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 [27] 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. [44, 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

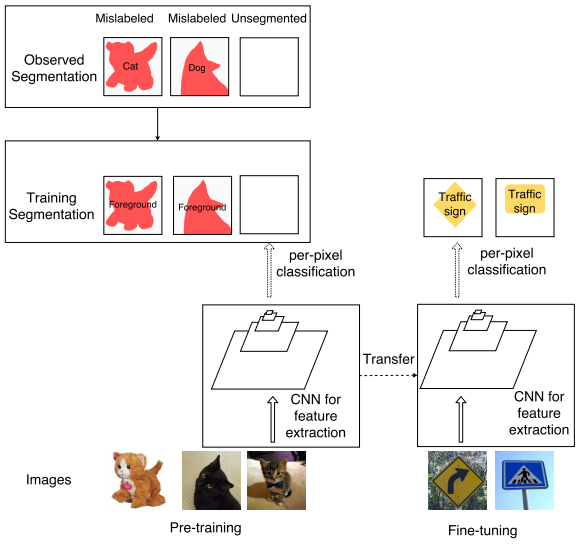

Figure 1: Learning representations with segmentation datasets that potentially contains mislabeled objects and missing segmentation. The observed segmentations may contain mislabeled segments and unsegmented objects. We propose (1) to train foreground/background segmentation instead of per class segmentation, (2) to apply a sigmoid loss for the background class in the presence of unsegmented objects. The learned representations can be used as weights initialization for another dataset of interest, e.g., traffic sign segmentation.

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 [25, 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 [29]. 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[30]. 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 [25] 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 [43]. 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 [24]. 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 the learned representations.

2. We propose to learn representations by training foreground/background segmentations instead of by training per-class segmentation.

3. We propose a class-dependent sigmoid loss to train deep neural networks with only positive and unlabeled data.

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. The proposed sigmoid loss is evaluated, compared to the class-weighted loss, for classification with positive and unlabeled data and segmentation with positive and unlabeled data in Section 5.1 and Section 5.2 respectively. 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. 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 [36], but also to other applications like object detection [11], and semantic segmentation[27]. 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. [31] Yosinski et al. [43] 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 [41, 28], deep belief networks [16, 22]. 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 [38, 32] 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. [34] 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. [38] introduced a linear noise layer on top of the model output, and Patrini et al. [32] proposed two forms of loss correction concerning the label observation bias. Xiao et al. [42] 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 [33] 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 [26, 24] 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 [23]. 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. [40] Du et al. [3] illus-

trated 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 [40] 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[40] 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.

Learning with only positive and unlabeled data Consider a dataset containing N training examples $(x_i, y_i, s_i), i \in \{1, 2, \ldots, N\}$, 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:

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

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. The problem of training classifier with the P set and the U set only is referred to as **PU learning**.

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 $\tilde{y_i}$ 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

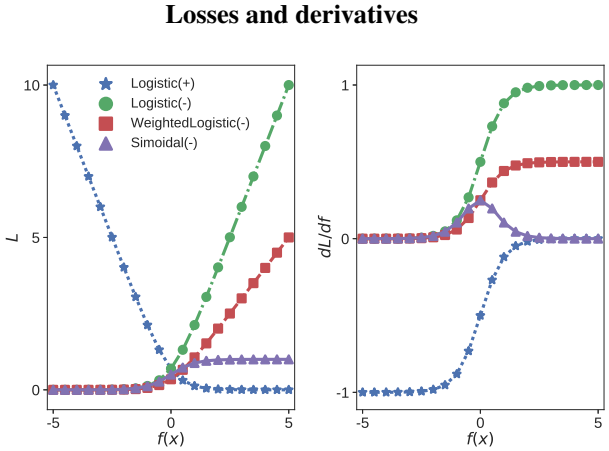

Figure 2: The differences in losses (left figure) and derivatives (right figure) with respect to the model f between the weighted logistic loss and the sigmoid loss for the negative class. The x-axis denotes the model output $f(x)$, varying from negative infinity to positive infinity. We present only $(-5, 5)$ 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.

is referred to as the **class-weighted loss** in the rest of this thesis. Empirically, the choice of α, β can be made based on the 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:

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

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:

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

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 sigmoid loss reaches a plaetau when the model output $f(x)$ grows above 2, and its derivatives with respect to the model f drops zero. By contrast, the weighted logistic loss, with $\alpha = 1$ and $\beta = 0.5$, for the negative class is scaled by a factor of 0.5 compared to the normal logistic loss for the negative class. It keeps growing with a rate of 0.5 as $f(x)$ grows in the large $f(x)$ area. A large output $f(x)$ represents that the model is more confident for the corresponding example being positive. The sigmoid loss follows our idea of not punishing highly confident positive predictions more than positive predictions with less confidence, whereas the weighted loss does not.

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. [33] 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 $\hat{y} = \operatorname{argmax}_{j \in \{-1,+1\}} \sigma_j(f(x;\theta))$ 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 For all losses being used to learning with positive and unlabeled data, we reweighed the positive and negative class based on their occurrences in the observed labels to alleviate the influence of imbalance introduced by the unlabeled positive examples. We introduced the hard bootstrapping loss only after training with a class-weighted cross entropy loss for a few epochs because it relies on a nonrandom model for sufficiently reliable prediction $\hat{y}$.

5 Experiments

5.1 Learning with only positive and unlabeled samples

In this section, we apply the sigmoid/softmax loss for the negative class to train classifiers with only positive data and unlabeled data and compare with the class-weighted loss regarding the learned decision boundary, the achieved precision and recall.

5.1.1 2D non-linear dataset

To investigate the decision boundaries led by the sigmoid loss for the negative class, we trained a multilayer perceptron with a two-dimensional, non-linear separable dataset.

Dataset 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.

Experimental setup The multilayer perceptron contains two layers, with six neurons per layer. The normal logistic loss, the class-weighted logistic loss, and the class-dependent sigmoid loss were trained independently, and the result optimal 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 the positive class and the negative class in the weighted logistic loss were 1 and 0.5 respectively.

Decision boundary with large margin by the sigmoid loss If trained with the sigmoid loss, the decision boundary is distant from the positive cluster with a relatively large margin, whereas the decision boundary for the

weighted logistic loss is still closed to the positive examples, as shown in Figure 3. 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 examples near the decision boundary than examples far away, illustrated as marker sizes in Figure 4. The derivative determines the update rate for the model weights. Higher derivative means a higher rate of update. Therefore, examples near the decision boundary have a higher contribution to the model updates for the sigmoid loss. In other words, the sigmoid loss emphasizes the positive predictions with low confidence. The emphasization of the uncertain predictions near the decision boundary for the sigmoid loss leads to a decision boundary converges to low-density regions for the inputs distribution. By contrast, the logistic loss has higher derivative for positive predictions with high confidence. It emphasizes the incorrect predictions with confidence so that the decision boundary is pulled toward the positive cluster by mislabeled positive samples. Since the sigmoid loss converges to an optimal decision boundary that has a relatively large margin from the positive cluster while still keeping distant from the negative samples, it is expected to achieve high recall and not sacrifice precision.

5.1.2 CIFAR dataset

To compare the precision and recall achieved by the class-dependent sigmoid/softmax loss and the class-weighted loss, we trained a CNN model to classify images of multiple relevant categories from non-relevant images with partially labeled relevant images.

Dataset We combined the CIFAR10 dataset and CIFAR100 dataset [19] to form a dataset with images for eleven classes: ten relevant classes from CIFAR10 and one non-relevant class for all categories from CIFAR100. Only part of the relevant images are labeled (with correct classes), and the rest of the relevant images forms an unlabeled (U) set together with the non-relevant images. Images from the unlabeled set were assigned negative labels.

Experimental setup 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 Table 6 in Appendix C.1. 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.

Higher recall and comparable precision with the softmax loss 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. The softmax loss for the non-relevant class can achieve higher recall without sacrificing the precision by much, compared to the class-weighted loss.

To compare the class-dependent softmax loss and the class-weighted cross entropy with varying percentage of labeled relevant images, we also trained models with datasets containing varing percentages of labeled relevant images. Figure 5 demonstrates that the class-dependent softmax loss performs slightly better than the

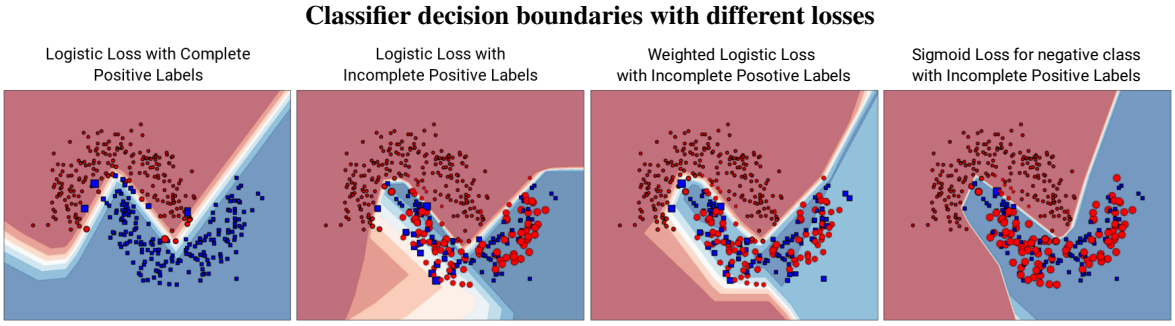

Figure 3: Decision boundaries of a 2-layer multilayer perceptron trained with different losses on a 2D moons dataset with the unlabeled positive. A **red circle** represents an example labeled as positive and a **blue square** represents the example has a negative label. The **background colors** indicate 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 large margin from the positive cluster as well as from the negative clusters. It is expected to achieve both high recall and high precision. (Best viewed in color.)

class-weighted cross-entropy when the percentage of labeled relevant images is neither too high (> 0.8) nor too low (< 0.2). When the percentage of labeled relevant images is high or low, the softmax loss behaves no worse than the class-weighted cross-entropy. Therefore, the sigmoid loss is in general better than weighting the losses for different classes.

5.2 Learning with incomplete segmentations

To compare the class-dependent sigmoid loss with the class-weighted logistic loss for training foreground/background segmentation with incomplete segmentations, we constructed an incompletely labeled dataset from the PASCAL VOC2011 dataset [10] with extra segmentations [13].

Dataset Objects from the 20 foreground categories of the PASCAL VOC2011 dataset were labeled as foreground, constructing binary segmentations with the background pixels. We selected objects in the training images completed at random with a probability of 0.5 to be labeled. The other objects and the background pixels were left unlabeled. Only single-object images were used for training and testing to avoid the influence of two adjacent objects joining as one object because of binary segmentation, 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 A Fully Convolutional Networks with AlexNet model (FCN-AlexNet), as shown in Figure 10 in Appendix A.2, was used for experiments because of its relatively small capacity and thus short training time. The FCN-AlexNet model was trained together with the normal logistic loss, the class-weighted logistic loss, and the class-dependent sigmoid loss independently to predict binary segmentation, determining whether a pixel is a foreground or background. The learned models were evaluated with the test set of the PASCAL VOC2011 dataset with complete binary segmentations. Weights from the pre-trained AlexNet model [20] were used as initialization for compatible weights of the FCN-AlexNet model. The other weights of the FCN-AlexNet model were randomly initialized with Xavier initialization [12]. The default hyperparameters of FCN-AlexNet in [27] were kept un-

Derivatives of different losses and decision boundaries

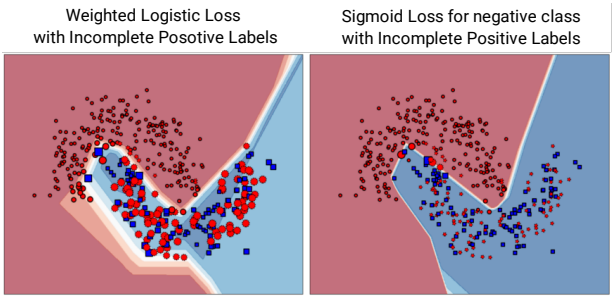

Figure 4: Derivatives w.r.t the last layer output for the two losses (normalized per class and shown as the marker size). The sigmoid loss has small derivatives for samples farther from the decision boundary and large derivatives for samples near the decision boundary, which is opposite to the weighted logistic loss. Higher derivative means the example has a higher rate to update the model weights during optimization. The sigmoid loss emphasizes the uncertain incorrect predictions (points near the decision boundary) in training, whereas the weighted logistic loss emphasizes the confident incorrect predictions (points distant from the decision boundary). (Best viewed in color.)

Classification performance with partially labeled relevant data

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 for the cross-entropy loss by a factor of 0.5 improves the mean f1-score significantly. The class-dependent softmax loss achieves higher recall than the class-weighted loss without sacrificing precision, but not as good as training with a set of labeled negative examples (R+N and 50%(R+N)). The class-dependent softmax loss achieves better f1-score than the class-weighted loss and is comparable to a state-of-the-art method, the hard bootstrapping loss.

changed. 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, and the highest mean IU achieved on the test set for the last five snapshots were summarized in Table 2.

Higher recall with the sigmoid loss As shown in Table 2, the class dependent sigmoid loss achieves the higher mean recall by approximately 0.07 than training with the normal logistic loss, and by 0.04 than the class-weighted loss. Specifically for the two classes, the foregorund recall class increases whereas the background recall decreases for the sigmoid loss. This difference in classes lead to a mean IU for the class-dependent sigmoid loss no better than the class-weigthed loss because the mean IU counts for both low false positive rate and low false negative rate. The class-dependent sigmoid loss improves the recall averaged for the foreground class and background class when training with incomplete segmentations.

Selective predictions made by the models trained with the sigmoid loss and the cross entropy loss were presented in Figure 6. For the two example images shown, the model trained with the cross entropy loss failed to segment objects from images whereas the sigmoid 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 than training with the sigmoid loss and incomplete segmentations. There is no example observed correctly

Classification performance with varying percentage of labeled relevant images

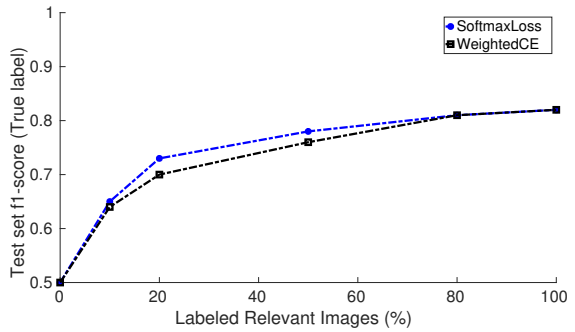

Figure 5: Comparing f1-score for the class-dependent softmax loss and the class-weighted cross entropy with varying percentage of relevant images labeled. The class-dependent softmax loss achieves better test f1-score than the class-weighted cross entropy when 20% and 50% percentage of relevant images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

segmented by the model with the class-weighted logistic loss but not by the model with the class-dependent sigmoid loss. These two examples show that the sigmoid loss improves the segmentation performance by segment a few more objects than the weighted logistic loss.

5.3 Learning representaions in the presence of mislabeled segmentations

To learn representations in the presence of lablel errors, we set up three experiments for three types of label errors, (1) inexhaustive segmentations, (2) objects mislabelling and (3) false positive segmentations, independently with three datasets constructed from a well-annotated dataset, the PASCAL VOC2011 segmentation dataset [10].

5.3.1 Datasets

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*. Three types of label errors of interest were introduced independently with stochastical corruptions to

Segmentation performance

Annotation	Loss	overall acc.	mean rec.	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 achieves higher recall than the class-weighted logistic loss and a similar mean IU as the class-weighted logistic loss.

the well-annotated pre-training dataset. To avoid the influence of the choice of the pre-training and fine-tuning splitting for categories, we divided the 20 categories of VOC2011 equally into four folds. The exact folds of categories are:

Fold 1 aeroplane, bicycle, bird, boat, bottle

Fold 2 bus, car, cat, chair, cow

Fold 3 dining table, dog, horse, motorbike, person

Fold 4 potted plant, sheep. sofa, train, TV

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. The original images were subsampled by four times to accelerate the training process.

5.3.2 Experimental setup

A Fully Convolutional Network with AlexNet (FCN-AlexNet) model [27], as shown in Table 10 in Appendix A.2, was used for segmentation. Models were first pre-trained with the pre-training datasets and then fine-tuned with the fine-tuning datasets. The fine-tuned models were

Fine-tuning performance of representations trained in the presence of random labels

	Pre-trained Models	Fine-tuning mean IU per pretraining-finetuning fold				Average mean IU
		Fold1	Fold2	Fold3	Fold4	
Baseline	RandomWeights	0.29 ± 0.01	0.29 ± 0.03	0.27 ± 0.01	0.30 ± 0.02	0.29 ± 0.02
	TrueLabels	0.29 ± 0.01	0.36 ± 0.01	0.29 ± 0.01	0.37 ± 0.01	$\mathbf{0.33 \pm 0.01}$
Objects Mislabelling	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	$\mathbf{0.32 \pm 0.02}$

Table 3: Segmentation performance for FCN-AlexNet models pre-trained on 15 categories from the PASCAL VOC2011 dataset and fine-tuned on the other five categories. The splits of pre-training and fine-tuning categories are organized in four folds. **RandomWeights** represents the randomly initialized weights; **TrueLabels** stands for the model pre-trained with true labels; **AllRandomLabels** denotes the model pre-trained with all random foreground labels; **HalfRandomLabels** is the model pre-trained with half random and half correct foreground labels; **BinaryLabels** demonstrates that the model is pre-trained with binary (foreground and background) segmentations; Random foreground labels for segmentations in the training set decreased the fine-tuning performance for the learned representations, compared to the true foreground labels. Training foreground and background segmentation instead of per foreground class segmentation improves the fine-tuning performance when the pre-training dataset contains mislabeled objects from one foreground class to another.

Example predictions

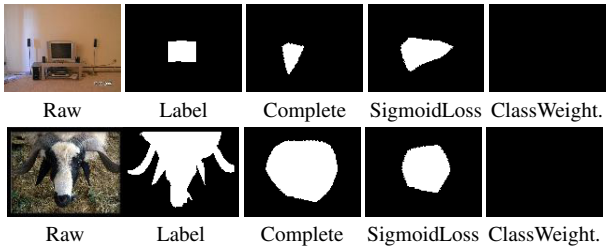

Figure 6: Example predictions made by models trained with the logistic loss and the class-dependent sigmoid loss. This figure presents two selective images for which the model trained with the logistic loss failed to segment objects, whereas the model trained with the class-dependent sigmoid negative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

evaluated by mean intersection over union ratio (mean IU) on the fine-tuning test set, which is referred 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. Random weights initialization were considered as the baseline. A well pre-trained model should at least outperform random weights initialization. The default hyperparameters of FCN-AlexNet in [27] 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.

5.3.3 Results

Training binary segmentations in the presence of objects mislabelling Objects Mislabelling denotes that a subset of segmented objects are mislabeled from one category to another in a training segmentation dataset. To validate that training binary segmentation can learn representations better than training per-category segmentation in the presence of objects mislabelling, we constructed three different datasets with (1) all random labels for objects, (2) half random and half correct labels for objects, and (3) all correct labels for objects. The learned representations with these three datasets were fine-tuned to segment the five fine-tuning categories and evaluated by a

Fine-tuning performance of representations trained in the presence of incomplete segmentations

	Pre-trained Models	Fine-tuning mean IU per pretraining-finetuning fold				Average mean IU
		Fold1	Fold2	Fold3	Fold4	
Baseline	RandomWeights	0.29 ± 0.01	0.29 ± 0.03	0.27 ± 0.01	0.30 ± 0.02	0.29 ± 0.02
	CompleteLabels	0.29 ± 0.01	0.36 ± 0.01	0.29 ± 0.01	0.37 ± 0.01	$\mathbf{0.33 \pm 0.01}$
Inexaustive Segmention	HalfUnsegmented	0.26 ± 0.01	0.30 ± 0.03	0.28 ± 0.03	0.32 ± 0.02	0.29 ± 0.02
	SigmoidalLoss	0.30 ± 0.01	0.37 ± 0.01	0.31 ± 0.02	0.34 ± 0.02	$\mathbf{0.33 \pm 0.02}$

Table 4: Segmentation performance for FCN-AlexNet models pre-trained on 15 categories from the PASCAL VOC2011 dataset and fine-tuned on the other five categories. The splits of pre-training and fine-tuning categories are organized in four folds. **RandomWeights** represents the randomly initialized weights; **CompleteLabels** stands for the model pre-trained with complete segmentations; **HalfUnsegmented** denotes the model pre-trained with half of the objects unsegmented; **SigmoidLoss** means that the model pre-trained with half of the objects unsegmented and with the sigmoid loss applied to the background class. Applying the sigmoid loss to the negative class when pre-trained with inexhaustive segmentations achieves a fine-tuning performance comparable to pre-training with the complete segmentations, better than training with the normal logistic loss.

Fine-tuning performance of representations trained in the presence of false positive segmentations

	Pre-trained Models	Fine-tuning mean IU per pretraining-finetuning fold				Average mean IU
		Fold1	Fold2	Fold3	Fold4	
Baseline	RandomWeights	0.29 ± 0.01	0.29 ± 0.03	0.27 ± 0.01	0.30 ± 0.02	0.29 ± 0.02
	NoFalsePositive	0.26 ± 0.01	0.37 ± 0.03	0.27 ± 0.01	0.33 ± 0.04	$\mathbf{0.31 \pm 0.02}$
False positive segmentaion	HalfFalsePositive	0.27 ± 0.01	0.34 ± 0.01	0.30 ± 0.01	0.32 ± 0.01	$\mathbf{0.31 \pm 0.01}$

Table 5: Segmentation performance for FCN-AlexNet models pre-trained on 15 categories from the PASCAL VOC2011 dataset and fine-tuned on the other five categories. The splits of pre-training and fine-tuning categories are organized in four folds. **RandomWeights** represents the randomly initialized weights; **NoFalsePositive** denotes the model pre-trained with no segmented object from the non-relevant categories; **HalfFalsePositive** represents the model pre-trained with segmented objects from the noninterested categories; Including the false positive segmentations in pre-training achieves no worse fine-tuning performance than not including the false positive segmentations, better than random initialization.

test set for the five fine-tuning classes.

Results in Table 3 suggests that pre-training with mislabeled foreground objects have a negative influence on the learned representations. Compared to the model trained with true labels, both models trained with all random labels and half-true half-random foreground labels do not present improvement of segmentation performance to the random weights initialization on the test set of the fine-tuning dataset.

We instead discard the labels for the objects and train binary (foreground/background) segmentation. The result representations achieve a fine-tuning performance better than training with the mislabeled objects and equivalent to the model pre-trained 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. This observation indicates that binarizing segmentation labels into foreground and background have little influence on the learned representations.

Categorizing the foreground classes We then investigate the influence of categorizing the foreground classes instead of grouping all of them as one foreground class.

Fine-tuning performance of representations trained with categorized segmentation labels

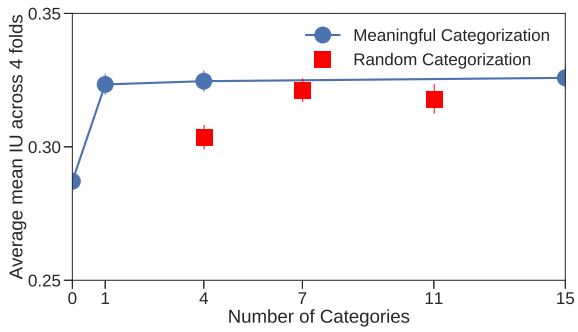

Figure 7: Test performance for the fine-tuned models pre-trained with varying categorization of pre-training classes. Zero categorize means no pre-trained weights used and the model was random initialized. Error bars located on lines denote the meaningful categorization, and isolated error bars denote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

We categorized the fifteen pre-training classes into four meaningful categories: person, animal, vehicle, indoor according to [10], and trained segmentation models to transfer. The fifteen pre-training classes were also randomly categorized into 4, 7, 11 categories, respectively, to pre-train segmentation models. The learned multi-categories segmentation models are then fine-tuned with the 5-categories fine-tuning dataset and shown as the error bars in Figure 7.

The line in Figure 7 demonstrates that the segmentations labeled in one category, in four categories, and in fifteen categories have no significant influence on the learned representations. All the three learned representations improve the fine-tuning performance compared to a random weights initialization show as the blue circle at categories=0. Additionally, the isolated error bars in figure 7 reveal that even training by random categorization of the foreground classes has little effect on the fine-tuning performance for the learned representations. This observation indicates that it is not necessary to group the foreground classes into meaningful categories to reserve the class specific information. Therefore, we propose to train foreground/background segmentations when the purpose is to learn representations that can transfer to another dataset.

Training with the sigmoid loss applied to the background class in the presence of inexhaustive segmentation Inexhaustive segmentation means that there exist unsegmented objects of interest in the training images. To validate the use of sigmoid loss for the background class can alleviate the negative effects of inexhaustive segmentation to the learned representation, we constructed a training dataset with only 50% of the objects segmented, together with a training dataset with 100% of the objects segmented. Segmented means the object is labeled as foreground, and unsegmented means the object is labeled as background. The class-dependent sigmoid loss and the normal logistic loss were applied to the dataset with 50% of the objects unsegmented. The learned representations were evaluated by fine-tuned and validated with the five-categories fine-tuning dataset.

The segmentation performance, mean IU, for the model pre-trained with incomplete segmentations is worse than the model pre-trained with complete segmentations by 0.04 when using the logistic loss to train, are demonstrated in Table 4. By applying the class-dependent sigmoid loss to pre-train models with half of the objects unsegmented, the learned representations achieve a fine-tuning performance comparable to the model pre-trained with complete segmentation. The representations learned with the class-dependent sigmoid loss is demonstrated to achieve better fine-tuning performance than the normal logistic loss.

Including false positive segmentations for training if they present False positive segmentations represents those segmented objects that are semantically meaningful but not from a pre-defined category to segment. To investigate the influence of including false positive segmentations for training, we consider a dataset contains dogs as the only category to segment. Objects from the other fourteen categories are not supposed to be segmented for

an error-free dataset without false positive segmentations. The model trained with this correctly labeled dataset is named as the NoFalsePositive model in Table 5. Another dataset, containing half of the objects from the other fourteen categories segmented, is referred to as the HalfFalsePositive model.

We observe, as presented in Table 5, that transferring the HalfFalsePositive model performs almost the same as the NoFalsePositive model and better than the random weights initialization. Based on this observation, we conclude that including the false positive segmentations for training have little negative effects on the learned CNN representations.

6 Discussion

Datasets with not only label errors but also segmentation errors In practice, datasets may also contain segmentation errors such as imprecise boundaries, oversegmenting and under segmenting the objects. Our proposed method of learning representation with segmentation data do not take these types of label noises into account. The investigation of the influence of segmentation noises on the learned representations is left for future works.

Disadvantages for the sigmoid loss First of all, we are not able to determine what is the threshold when the sigmoid loss saturates. 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 $\alpha(\frac{1}{1+\exp{(\beta z)}})^{\gamma}$, 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. Future investigation for this parametrized general sigmoid loss and the corresponding choices of the hyperparmeters is required to achieve potentially better classification performance.

Secondly, a loss saturates in the regions of confident predictions can have its 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 negative labels percentage.

The difference between learning with positive and unlabeled data and learning with incomplete segmentations In Section 1, we argue that learning in the presence of unlabeled foreground pixels is similar to learning with positive and unlabeled data. Despite the similarities between the two, there is also a difference between learning with unlabeled foreground pixels and learning with positive and unlabeled examples. Each example in the normal PU learning setup is independent of each other, whereas the pixels in images are not. In practice, there is often a spatial dependence for pixels of being labeled or unlabeled. When we apply the sigmoid loss to the background class for learning segmentation moels with incomplete segmentations, we assume that the probability for a foreground pixel of being labeled as the background is independent of its neighbor pixels. A method to interpret the spatial dependence for pixels of being labeled in the model can potentially further improve the segmentation performance.

7 Conclusion

We investigate in this paper to learn representations by training with segmentation datasets containing label noises. (1) We report both mislabeled objects and unsegmented objects in a segmentation dataset negatively influence the transferability of the learned representations, i.e., how well the representations transfer to another dataset. By contrast, false positive segmentations, i.e., segmented objects that are not supposed to be segmented, do not influence the learned representation as significant as the other two types noises do. (2) We present that training foreground/background segmentation can produce learned representations comparable to the representations trained with per-class segmentation. In addition,

binarizing classes for segmentation alleviates the negative influence on the learned representations introduced by the mislabeled objects. (3) We propose a class-dependent sigmoid loss to not over-punish the confident, positive predictions for the negative class when there exist poorly labeled negative samples. 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 segmentation performance and the transferability of the learned representations for a dataset with 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. Learning multiple layers of features from tiny images. 2009.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

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

[35] 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.

[36] 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.

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

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

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

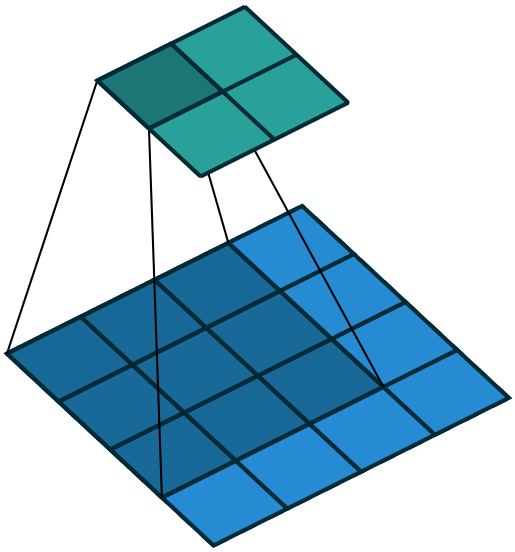

Figure 8: A basic convolution operation. A 3x3 convolutional filter convolves with a 3x3 window sliding over the image (bottom). The output of convolutions at each sliding position form a feature map (top). This figure was drawn by Dumoulin and Visin [6]

Example classification model: FCN-AlexNet

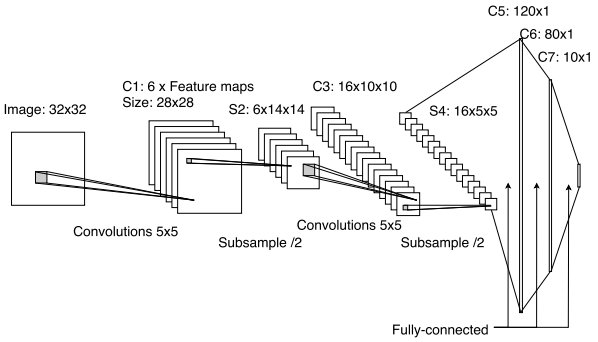

Figure 9: An example convolutional neural network, LeNet-5 [21]. The first convolutional layer of LeNet contains 6 convolutional kernels of size 5x5 and each convolutional kernels convolve with small windows sliding over the images and produce a feature map of size 28x28. Each output in the produced 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 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.

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. For example, LeNet-5 (1998) [21], a simple CNN model for handwritten digit recognition, is shown in Figure 9.

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. [21] Different from the traditional handcrafted features, learnable convolutional features normally generalize well and can achieve better performance for dataset with a complex input distribution. [20] 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 [35], Microsoft COCO [25], 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. [27] 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. [27] 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.

Example segmentation model: FCN-AlexNet

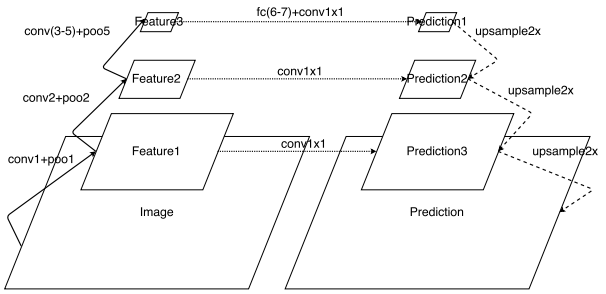

Figure 10: Fully convolutional network with AlexNet (FCN-AlexNet) by Long et al. (2015) [27]. Features of different resolutions are stacked in a feature pyramid on the left-hand side, with the image at the bottom of the pyramid. Predictions of different resolutions are piled in a prediction pyramid on 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 upsampling 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 Back-propagation

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:

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

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 6: 8-layer Convolutional Neural Networks used in the classification of the CIFAR dataset.

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:

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

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 6.

C.2 Evaluation metrics

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

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

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

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

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