Preface

The thesis report contains a preface that explains the topic of the thesis, the context (institute or company), the main findings in a few lines and the names of the members of the thesis committee. The preface may end with a few acknowledgements, and completed with name and date.

The main goal of this thesis is to study if, in a transfer learning scenario, the transferability of pre-trained weights is affected by the existence of segmentation noises for training examples and to explore methods compensating the negative effects if exist. Transfer learning is relevant when segmentations in a domain of interest are difficult to obtain on a large scale. The transferred model is often a classification model trained with a selective subset of images from ImageNet. Another choice of transferred model is a segmentation model trained with another segmentation dataset to pursue more transferable weights. However, various noises may occur in segmentations if not enormous effects were made to correct them. Being able to learn comparable transferable weights even in the presence of these noises can be beneficial to save efforts made to correct every single segmenting error and create more segmentations using the saved efforts. This can be helpful in collecting segmentation datasets on a large scale with less effort and money cost when using the crowd-sourcing power.

We found that mis-segmentation noises had less influence on weights transferability compared to misclassification noises and inexhaustive segmentations. We proposed to categorize or binarize classes so that misclassified labels would not have as much as effects on weights transferability as training with the exact classes. A modification to cross entropy loss was proposed to alleviate the negative influence of unlabeled positive examples.

Members of the thesis committee include Prof. dr. A.Hanjalic (Multimedia Computing Group, TU Delft) as the chair, dr. J.C. van Gemert (Vision Lab, TU Delft) who was the daily supervisor of the student, and Prof.dr. M. Loog (Pattern Recognition Laboratory, TU Delft) and dr. Z. Szlvik (CAS Benelux, IBM).

I sincerely appreciate the magnificent supports provided by dr. J.C. van Gemert, Prof.dr. M. Loog and dr. Z. Szlvik as co-supervisors day to day. I would also like to thank dr. D.M.J. Tax for his expert knowledge in the domain.

Jihong Ju
August 24, 2017

Learn transferable representations in the presence of noisy labels for segmentation

Jihong Ju
Faculty of Electrical Engineering, Mathematics and Computer Science
Mekelweg 4, 2628 CD Delft
j.ju@student.tudelft.nl

August 24, 2017

Abstract

Transferring pre-trained representations is widely adopted by convolutional neural networks for semantic segmentation because the training data is often available only on a small scale. In domains where adapting convolutional networks for object recognition is difficult, we propose to pre-train segmentation models with datasets containing mislabeled segmentations and unsegmented objects. Our experiments demonstrate that both mislabeled segments and incomplete segmentation lower the fine-tuning performance of the learned representations. To get rid of the negative effect of objects label noises, we propose to assign objects of any categories a foreground label instead of the specific object categories. Learning representations by segmenting foreground and background turns out to improve the fine-tuning performance significantly when label noises are dominant in the pre-training data. In the existence of unsegmented objects, a sigmoid loss for the background class is proposed to balance precision and recall more effectively than simply weighting classes. The proposed class dependent, sigmoid loss achieves both better pre-training and better fine-tuning performances than the class weighted loss in the presence of incomplete segmentation.

1 Introduction

The often limited availability of training samples motivates most state-or-the-art deep learning based segmentation models [28, 2, 15] to transfer convolutional neural network (CNN) models [20, 37, 39, 16] trained on a subset of images from ImageNet. Compared to object recognition tasks, it is tougher to collect a dataset for semantic segmentation on a large scale. 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 [26], contains 123,287 images of 80 object categories, smaller in the number of images than the ILSRVC dataset by a factor of 10 approximately. Transferring weights from the pre-trained ImageNet models can provide a segmentation performance boost in the limitation of lacking training samples, as reported in [28] and adopted by [2, 15].

However, it can be challenging to employ directly the ImageNet CNN models for semantic segmentation. Firstly, the first layer of ImageNet models is not compatible with RGB-D images and 3D images like CT scans and MRI scans in 3D. [43]. Secondly, ImageNet models were originally trained with natural images at relatively low resolution. In some domains of interest, images can be non-natural, such as aerial images, images from bird's eye view, and medical images; In some other domains, images may have different lighting conditions or have a high resolution than the ImageNet ones. Lastly, segmentation models do not necessarily follow the architecture design of classification models. For example, the object recognition models pursue features invariance to better capture semantics regardless the variations in objects. The re-

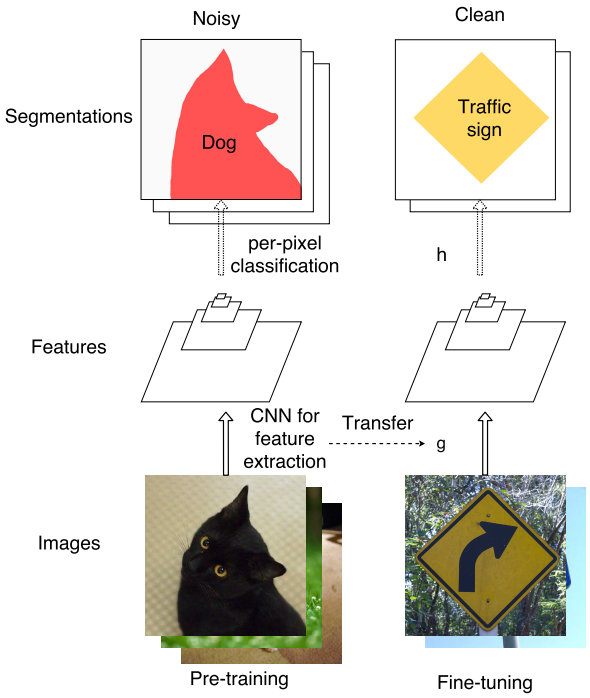


Figure 1: Transferring feature extractor(g) from convolutional neural networks for semantic segmentation pre-trained by potentially incomplete, mislabeled segmentations.

sult translation invariant and resolution-reduced features could reduce the localization accuracy which is not essential for object recognition but is critical for object segmentation. [43, 2] This difference in requirements of features can result in different neural network architectures for segmentation models Therefore, re-training ImageNet models, or even pre-train models from scratch in particular cases, using a segmentation dataset similar to but different from the one of interest can be more beneficial than employing ImageNet models directly.

The segmentation datasets for pre-training may contain label noises. 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 [26, 10], to ensure the correctness of segmentations. If not requiring "gold standard" segmentations for training, the efforts saved for correctness can be made to segment more images for a larger dataset. Besides, some labels other than the manual ones may be freely available for particular tasks. But these labels often contain structural noises depending on the way they were created. For example, digital maps, like OpenStreetMap, can be used to segment aerial images. Segmentations constructed from the maps could suffer from incompleteness as well as registration problems. [30]

Noises of different kinds can exist in segmentation labels. In particular, we consider mislabeling errors for the whole segment, assuming the outline of objects is always correct. We categorized object mislabelling into three categories: inexhaustive segmentation, misclassification, and false segmentation. **Misclassification** of objects from one category to another exist occasionally even for well-annotated datasets. For example, the Microsoft COCO dataset [26] contains some misclassified bears and teddy bears 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 segmentation** denotes that semantically meaningful objects from an undefined category are wrongly segmented, as objects of interest. It may occur due to the unclear definition of categories, visual similarities between objects, etc.

We simulated these three types of noises with a well-annotated dataset separately and studied their influences to the trained weights transferability in a transfer learning setup demonstrated in Figure 1. Ideally, noises in the pre-training datasets should not significantly affect transferring the learned model to another dataset. But since label noises were proved to result in worse classificatin performance [38, 32], it could also negatively influence the model transferability. If negative influences introduced by label noises are remarkable, methods to compensate the noises become necessary.

Supposing misclassification had a negative influence on feature transferability, as we will show in Section

5.1, simply segmenting the foreground objects against the background can produce correct labels for misclassified objects. It can be regarded as converting precise but potentially inaccurate labels to accurate but imprecise labels. Training transferable features may not require as precise supervision as training classifiers because transferability of features are correlated to feature generality, i.e., if they are independent of particular categories [42]. Besides, Jain et al. [19] proved the possibility of training a fully convolutional network to generate foreground segmentations. Surprisingly, the model trained on over 1 million images generalizes well to object categories not included in the training. Therefore, we argue that it should be possible to learn transferable representations by segmenting foreground and background in images.

If we consider inexhaustive segmentation only, 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. The P set in an incompletely segmented dataset is comprised of segmented pixels while the rest of the pixels construct the U set. The main characteristic of the U set is no easy way to generate reliable negative labels out of it. Semi-supervised learning techniques are consequently not applicable as a result of the absence of negative training samples. The set of background pixels containing unsegmented objects can fulfill this property of U set. Learning to segment objects in the presence of inexhaustive segmentation can be therefore considered as 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 modify the loss for the negative class accordingly, following a branch of previous studies for PU learning as discussed in Section 2.

The main topics discussed in this thesis are:

1. How mislabeling of segmentations influences feature transferability of CNN models for semantic segmentation?

2. Compared to the inaccuracy of pre-training labels, how impreciseness of labels affects feature transferability?

3. How to modify the loss function to achieve better pre-training and fine-tuning performances in the presence of incomplete segmentations?

The rest of this thesis is organized as follows: In the next section, we summarize related works. In Section 3 we formulate the problem of transferring segmentation models pre-trained with noisy segmentations, and how to synthesize the three types segmentation noises. We introduce a sigmoid loss for the negative class to compensate noisy negative labels in Section 4. Experiments in Section 5.1 was designed to study the influences of inexhaustive segmentations, misclassification and false segmentation separately. The proposed sigmoid loss, together with the class-weighted loss and a modified hard bootstrapping loss, is evaluated in simulated PU learning setups in Section 5.2. 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 [36, 42] but also to other applications [12, 28]. Transferring weights of pre-trained CNN models allows improving the model performance without engaging in much efforts spent for data-labeling. [31] 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 even completely 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 can be expected to have no significant influence to the transferability of the result pre-trained models.

Apart from supervised pre-training, one can also perform unsupervised learning to obtain pre-trained features typically with auto-encoders [41, 29], deep belief networks [17, 22]. Though a few studies [9, 8, 1] discussed the advantage of unsupervised pre-trained features compared to random weights initialization, the distance between the two has been shortened ever since the arises of

modern initialization strategies, namely Xavier initialization [13] and its variants. We used random weights initialization as the lower baseline for pre-training with noisy labels. Features 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 effects of diluting instead of replacing correct labels with noisy labels and concluded that larger training set is of more importance than lower the level of noise. 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 feature robustness label noises.

Methods, including a linear noise layer on top of the model output [38], loss correctness with an estimation of the noise transition matrix [32] were proposed to compensate the negative effect on classification performance introduced by flipped labels. To study if the proposed methods can alleviate the influence of noisy labels as expected, both works synthesized the random flipped labels from clean labels. We designed our experiments using stochastically synthesized noisy segmentations from perfect segmentation as well.

Positive and Unlabeled Learning Traditional positive and unlabeled learning methods originally proposed for text classification [27, 24] do not extend well to deep learning models. These traditional methods often follow a two-step strategy: first identifying a set of reliable negative samples (RN set) from U set and then iteratively build a set of classifiers with RN set and P set, while updating the RN set with a selected classifier. It would take tremendously longer time to train a couple of deep learning models iteratively than to train a sequence of naïve Bayesian (NB) models or supported vector machines (SVMs). Therefore, it would be unrealistic to train CNN models in this manner.

Alternatively, one can treat all unlabeled examples as negative and simply reweigh positive and negative examples. [23] Under the assumption of randomly labeled positive examples, Elkan & Noto [7] demonstrated 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 but it is possible to extend the weighted logistic loss and train deep neural networks for multiple classes with only positive and unlabeled examples.

The problem of weighting negative examples is 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] illustrated that logistic loss and hinge loss perform worse than ramp loss in the PU classification setting due to their superfluous penalty for confident predictions. se different losses for positive and negative data. 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 gradient based optimization.

To avoid over-punish confident predictions unnecessarily, Tax & Wang [40] uses class-dependent, non-convex loss to train classifiers to retrieve small set of relevant objects from objects of a large, dominant class; Lin et al. [25] proposed a so-called Focal Loss to down-weight confident predictions and thus focus training on hard negatives for imbalanced learning. We proposed in Section 4 to use a sigmoid loss[40] for the negative class to alleviate superfluous punishment for confident, positive predictions. Additionally, Reed & Lee [33] proposed a bootstrapping loss to emphasize *perceptual consistency* in training when incomplete and noisy labels exist. It has a similar effect of reducing losses of confident predictions. We modified the hard bootstrapping loss to interpret the prior knowledge that positive labels are reliable.

3 Feature transferability with label noises

3.1 Problem Formulation

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:

$$f(I) = h(g(I))$$

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 $f(I)$:

$$f^* = \underset{f}{\operatorname{argmin}}\, L(S, f(I))$$

Feature Transferability A pre-trained feature extractor, g_t, can be transferred as an initialization of the feature extractor and is expected to result in a better performed f^* on the fine-tuning test set than a random choice of initialization g_0. The corresponding fine-tuning performance improvement indicates the transferability of the pre-trained feature extractor. Ideally, a feature extractor pre-trained with noisy labels would result in a fine-tuned model with equivalent performance as one pre-trained with true labels. The difference in the result fine-tuning performance improvement between the noisy and clean pre-trained feature extractor can tell the influence of label noises on feature transferability.

3.2 Label noise simulation

It is challenging to find a dataset with both clean and noisy labels available, so we tried to synthesize 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 $h \times w$:

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

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

Inexhaustive segmentation

Given the true labels, which pixels belong to the same object is known. Inexhaustive segmentation can be synthesized as 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: $1 - p_{0k}$.

Misclassification

To synthesize misclassification, we can stochastically convert pixel labels of segment for an object of the k-th category from k to j with probability p_{jk}, where $j, k \le K$ 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 segmentations

The presence of false segmentations can be synthesized 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.

4 Modifications to the cross-entropy loss for PU Learning

PU Learning A training set of the positive and negative (PU) learning problems contains only a set of positive examples (P set) and a set of unlabeled samples (U set). Unlabeled examples can be either positive or negative, meaning that there is no reliable negative examples available One straightforward way to generate negative examples for training is to treat all the unlabeled examples as a set of negative examples with noises. The problem then converts to learning with clean positive labels and noisy negative labels. The goal of solving a PU learning problem is to learn a classifier that predicts as many positives as possible while keeping the false positive rate low, regardless the influence of false negative labels. In other words, the

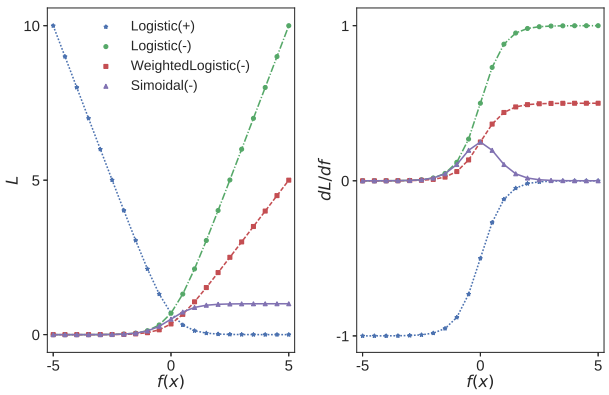


Figure 2: The differences in losses (top figure) and derivatives (bottom figure) with repect to logits between the weighted logistic negative loss and the sigmoid negative loss. The **+** sign represents the loss of positive samples and the **-** sign stands for the loss of negative samples. The sigmoid loss of negative examples reaches a plataeu and the derivative drops to zero in the very positive region, whereas the weighted logistic loss for negative is just a linearly scaled logistic loss.

purpose is to achieve high recall without sacrificing too much precision.

Weighted logistic loss for negative class The mislabeled negative samples bias the classifier to have low recall. It is possible to balance precision and recall by simply weighing the positive and negative examples differently, namely, let the positive and negative examples have different rates of contribution to the total loss. Suppose a logistic loss is used, the corresponding weighted loss for a input-output pair (x, y) for a classifier determined by parameters θ is:

$$l(x, y; \theta) = \begin{cases} -\alpha \log P(y = +1|x; \theta), & y = +1 \\ -\beta \log P(y = -1|x; \theta), & y = -1 \end{cases}$$

where α and β are weights for positive and negative class respectively, and $P(y|x; \theta) = \sigma(f(x; \theta))$ is the probablistic predictions by model $f(\cdot)$, activated by the sigmoid function $\sigma(\cdot)$. This loss is referred to as the **weighted loss** in the rest of paper. Empirically, the choice of p, q can be made based on the highest precision and recall achieved on a validation set, or alternatively based on a class priors estimation[5].

sigmoid Loss for negative class As motivated in Section 2, we used a class-dependent loss to down-weight the loss contribution of very positive predictions negative labels and still making full use of the clean positive labels. The loss of positive examples is still a normal logistic loss and the loss of negative examples is replaced with a sigmoid loss [40]:

$$l(x, y; \theta) = \begin{cases} -\alpha \log P(y = +1|x; \theta), & y = +1 \\ -\beta(1 - P(y = -1|x; \theta)), & y = -1 \end{cases}$$

We called this class-dependent loss **sigmoid loss** in a sense it uses a sigmoid function as the loss of negative samples.

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.

Hard bootstrapping loss for 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 a pair of inputs and label (x, y) is:

$$l(x, y; \theta) = \begin{cases} -\log P(y = +1|x; \theta), & y = +1 \\ -\beta \log P(y = -1|x; \theta) - (1 - \beta) \log P(y = \hat{y}|x; \theta), & y = -1 \end{cases}$$

where $\hat{y} = \text{argmax}_{j \in \{-1,+1\}} P(y = j|x)$ is the class with the highest predicted probability and $0 < \beta < 1$. The first term of the objective 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 **bootstrapping loss** for the rest of this paper.

Extend PU learning to multiclass problems For classification problems with multiple positive classes and one

negative class, the weighted negative loss, sigmoid negative loss and bootstrapping negative loss can be extended to train deep learning models in the presence of negatively labeled examples from various positive classes. The logistic loss naturally extends to the multi-class scenario as the sigmoid activation extends to the softmax function. Similar modifications can be then made to the cross-entropy loss, namely, keeping the losses for positive classes unchanged and applying the modifications to the loss for the negative class. We used the manner of extending the losses in this work. Alternatively, one can apply a one-vs-all strategy, with which the normal logistic loss is used for positive classes while the weighted, sigmoid and bootstrapping loss can be used for the negative class.

Extend PU learning to segmentation By assuming the probability of missing positive label for a pixel is independent of its neighbor pixels, the weighted negative loss, sigmoid negative loss and bootstrapping negative loss are all applicable to per-pixel classification problems if classification for each pixel is made independently.

Implementation details We introduced the sigmoid negative loss 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 The influence of simulated segmention noises on feature transferability

Experiment setup To evaluate the influence of inexhaustive segmentations, misclassifications and false segmentations on feature transferability, we set up experiments with simulated label noises from a well-annotated dataset, the PASCAL VOC2011 segmentation dataset [10]. 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 [28] for segmentation in the presence or absence of synthesized segmentation errors. The fine-tuning dataset was used to fine-tune the weights of convolutional layers from the pre-trained FCN-AlexNet models. The non-transferable layers of FCN-AlexNet were randomly initialized with Xavier Initialization. Inexhaustive segmentations, misclassifications, and false segmentations were simulated independently with stochastical corruptions to the well-annotated pre-training dataset, followed the descriptions in Section 3.1. 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.

Experiment details 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. Experiments run for each fold independently, and the exact partitions of each fold are listed in Table 1. The training dataset was enriched with extra segmentations by Hariharan et al. [14] 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. Fully Convolutional Networks with AlexNet was used for experiments because of its relatively small capacity and thus short training time. The existence of an ImageNet model for AlexNet was used to set a baseline of performance. 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 1. The default hyperparameters of FCN-AlexNet in [28] 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.

Inexaustive Segmentation Fine-tuning performance for pre-trained models with complete labels and incomplete segmentations are summarized in Table 1. Pre-training data with half of the objects unsegmented results in fine-tuned models with a worse average mean IU (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 weights transferability. In a positive and unlabeled (PU) learning setting, not only the classification performance but also the transferability of learned weights decreases due to the noises in negative labels.

Misclassifications The influence of random object labels on feature transferability is also demonstrated in Table 1. 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. Therefore, misclassification of objects in segmentations negatively impacts the transferability of CNN models in this simulated experiment setup.

Additionally, we binarized pre-training classes to foreground and background only, achieving a fine-tuning performance better than training with the exact 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 feature transferability than imprecise labels.

Precise labels vs. Accurate labels To validate that accurate imprecise labels can have little negative effect on feature transferability. In figure 3, we group pre-training classes into a various number of categories, not only two classes, to present the change of feature transferability when the preciseness of pre-training labels decreases. Addition to 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 also randomly categorized into 4, 7, 11 categories and shown as isolated error bars in figure 3 at categories=4, 7, 11 respectively. Figure 3 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).

Compared to the significant influence of randomized precise labels, the imprecise binarized or categorized labels is less harmful to image representation pre-training for semantic segmentation. Binarizing or grouping classes into hyper-categories can be helpful to learn better transferable weights when misclassification of objects is dominant.

False segmentations We also study the influence of incorrectly segmented objects from non-predefined but meaningful categories, the false segmentations, on feature transferability. To simulate false segmentations, 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 3.1. In the presence of false segmentations, in-

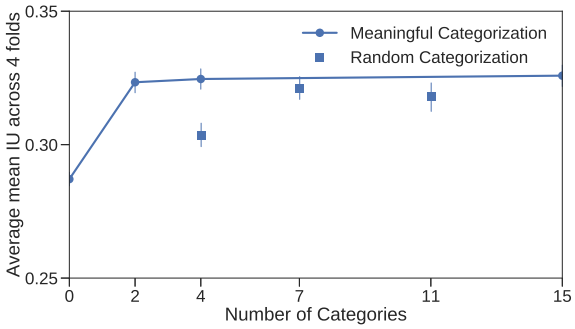


Figure 3: Test performance for fine-tuned models, varying categorization of pre-training classes to pre-train. Zero categorize means no pre-trained weights used and the model was random initialized. Error bars located on lines denote the meaningful categorization, and isolated error bars denote random categorizations (RC) of the 15 classes. The displayed mean IU/mean accuracies and standard deviations were averaged over four folds listed in Table 1. The line shows that binarizing and categorizing classes meaningfully had little negative effect on feature transferability.

stances 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 FalseSegmentHalf in Table 1. 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 NoFalseSegmented.

We observe from Table 1 that transferring the HalfFalseSegmented model performs almost the same as the NoFalseSegmentated model. Therefore, false segmenting objects from the 14 categories other than the cat (or dog) have a little negative impact on pre-training CNN models. The exact influence of false segmentations in practice, however, may depend on how similar the target and non-target categories are, or whether false segmentations are consistent, which requires further investigation.

5.2 Learning with only positive and unlabeled samples

PU classification for a 2D non-linear dataset To compare the weighted logistic loss and the sigmoid loss for negative class, we visualize the optimal decision boundaries for a two-layer multilayer perceptron, with six neurons per layer, achieved by the two losses respectively, with a toy 2D 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. The leftmost column in the figure shows the multilayer perceptron trained with complete positive labels whereas the other three columns show the classifier trained with half of the positive examples labeled as negative.

Weighted logistic loss vs. sigmoid loss for the negative class Figure 4 demonstrates the differences in optimal decision boundaries for classifiers trained by the logistic loss, the weighted logistic loss, and the sigmoid loss for negative class respectively. If training with the sigmoid loss, the mislabeled positive examples far away from the decision boundary do not contribute more losses than ones that are less distant from the decision boundary. The loss derivative with respect to the model weights is smaller for confident predictions in overall than for uncertain predictions, illustrated by marker sizes in Figure 5. As a consequence, training with the sigmoid loss emphasizes the negative examples predicted as positive with low confidence instead of equally down-weighting all negative examples. The result decision boundary for sigmoid loss becomes distant from the positive cluster with a clear margin. By contrast, the weighted logistic loss results in a decision boundary still in a similar shape as the normal logistic loss.

CIFAR dataset Images from the CIFAR10 dataset were used as positive (P) examples, and images from the CIFAR100 dataset were considered as the negative (N) examples. We trained a CNN model to classify images into eleven classes: ten positive classes from CIFAR10 and a negative class for images from CIFAR100. To synthesize a PU learning setup, we correctly labeled only part of positive images from CIFAR10 and the rest were assigned

	Initial Feature Extractor	Fine-tuning mean IU per pretraining-finetuning fold				Average mean IU across four folds
Fine-tuning categories		aeroplane, bicycle, bird, boat, bottle	bus, car, cat, chair, cow	dining table, dog, horse, motorbike, person	potted plant, sheep, sofa, train, TV	
Upper baseline& lower 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
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	$\mathbf{0.33 \pm 0.02}$
Misclassification	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	$\mathbf{0.32 \pm 0.02}$
False Segmentaion	NoFalseSegmented	0.26 ± 0.01	0.37 ± 0.03	0.27 ± 0.01	0.33 ± 0.04	$\mathbf{0.31 \pm 0.02}$
	FalseSegmentedHalf	0.27 ± 0.01	0.34 ± 0.01	0.30 ± 0.01	0.32 ± 0.01	$\mathbf{0.31 \pm 0.01}$

Table 1: 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 (upper baseline); **RandomWeights** indicates that the randomly initialized weights (lower baseline); 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 pre-training classes as foreground and background help overcome the negative effects of random labels. Including false segmentations in training (**FalseSegmentedHalf**) had the same fine-tuning performance as not including false segmentations (**NoFalseSegmented**), better than random initialization.

negative labels together with CIFAR100 images, forming the unlabeled (U) set. An eight layer CNN model was trained with different choices of losses and with the labeled positive examples and unlabeled examples. Model performances were then evaluated on a separate test set of combined CIFAR10 and CIFAR100 with true labels.

The architecture of the CNN model can be found in Appendix D. 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. Note that there is no category overlap between CIFAR10 dataset and CIFAR100 dataset.

Table 2 summarizes the test precisions and recalls of using different losses to learn with true positive labels and noisy negative labels, compared to training with the normal cross-entropy loss. With 50% of the positive example correctly labeled and the rest assigned negative 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 negative loss by a factor of 0.5, the model is balanced for precision and recall and the result f1-score is improved significantly. The sigmoid loss and hard bootstrapping loss achieves significantly better recall without sacrificing precision. The f1-scores achieved by the simoidal loss and hard bootstrapping loss are slightly better than the weighted negative loss, though not as good as training with clean labels with the same number of positive examples.

By varying the percentage of labeled positive examples, we compared learning with the labeled positive examples and noisy negative examples with the three differ-

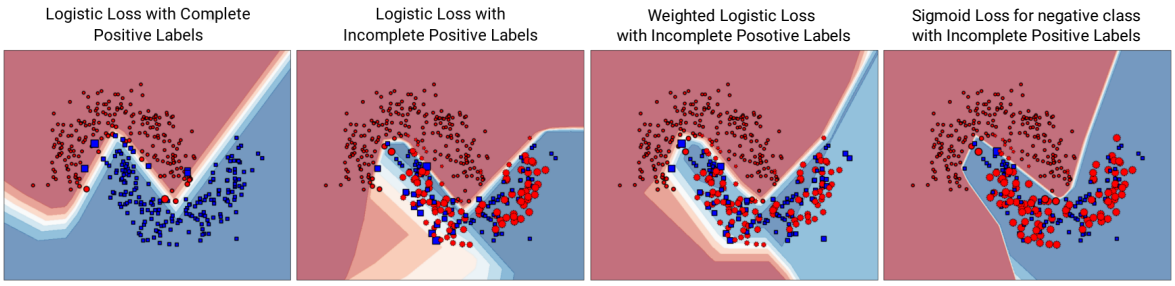


Figure 4: Decision boundaries of a 2-layer multilayer perceptron trained with different losses on a 2D moons dataset with the unlabeled positive. A **red circle** indicates an example labeled as positive and a **blue square** indicates the example has a negative label. The **background colors** represent the probability for the area to be positive given by the classifier trained with the given samples and labels: **red** for high probability areas, **blue** for low probability areas and **white** for the class transition areas, i.e.decision boundaries. The **markers sizes** demonstrates the per-class normalized training losses. Compared to the normal logistic loss and weighted logistic loss, the decision boundary optimized with sigmoid negative loss is more distant from the positive cluster. (Best view with color)

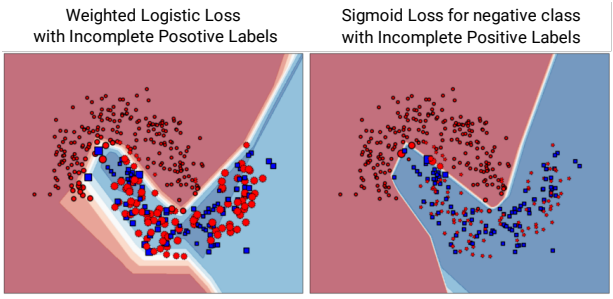


Figure 5: Normalized derivatives w.r.t the output per class (marker size). The sigmoid loss has small derivatives for predictions farther from the decision boundary. (Best view with color)

ent losses as shown in Figure 6. Similarly as a result at 50% positive examples labeled, the sigmoid negative loss and hard bootstrapping loss had only little improvement compared to the weighted negative loss. The assumption we made for the sigmoid negative loss was that the probability of a negative example being wrong is dependent on the confidence. However, the synthesized mislabeled positive examples were distributed at random when we synthesized them, meaning that the probability for a negative label being wrong is independent of the underlying distribution of the inputs. No matter where the decision boundary is, such uniformly distributed incorrect negative labels are independent of the prediction confidence. Therefore, it is expected for the sigmoid negative loss and bootstrapping loss to have similar performance as the weighted negative loss.

Figure 6 also demonstrates that learning with a subset of reliable negative examples (PN setup) outperformed learning with positive and unlabeled examples (PU setup) for any percentage of labeled positives and any losses being used. This result was expected because the PN setup delivers extra information about which images in the unlabeled set are negative. The PU setup is therefore only relevant when it is impossible to easily construct a subset of negative examples from the unlabeled data. Otherwise, training with true positive and negative labels, and potentially with semi-supervised learning, would be superior to training with positive and unlabeled examples only.

Pascal VOC2011 segmentation task To study the effect of sigmoid negative loss on compensating incomplete segmentations, we used again the PASCAL VOC2011 dataset with extra segmentation [14]. We synthesized inexhaustive segmentations the same way as described in Section 5.1. The same AlexNet-FCN model was trained together with the different loss functions for class 0 to

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

Table 2: Overall accuracy, mean precision, recall, and f1-score over classes on a test set of the CIFAR dataset with true labels. For each class, precision, recall and f1-score were measured by one-vs-all. CIFAR10 images were used as positive images (P), and CIFAR100 images were used as negative images (N). The **top three rows** summarize the upper baseline, middle baseline, and lower baseline for learning with positive and unlabeled examples. **Upper baseline** Model trained with the complete positive and negative labels; **Middle baseline** Model trained with 50% of the positive and negative examples; **Lower baseline** Model trained with 50% labeled positive examples and unlabeled examples; The proposed losses in the **bottom three rows** should achieve as high recall as possible while keeping the precision high as well. Both the sigmoid loss and hard bootstrpping loss for the negative class perform better than simply weighing the classes. The sigmoid loss for negative class achieves the highest mean recall, whereas the modified hard bootstraping loss has a higher accuracy with 50% positive examples unlabeled. Experiments were repeated three times and the standard deviation of results is approximately 0.01.

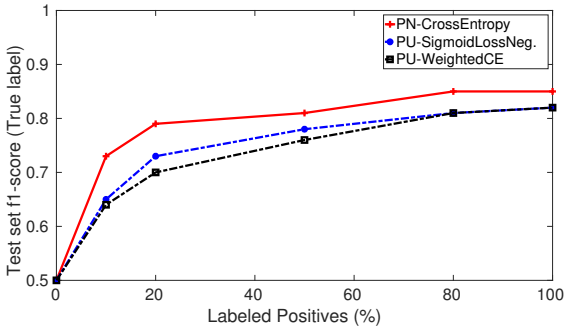


Figure 6: Comparing f1-score achieved by the weighted cross-entropy loss and the sigmoid loss for negative class by varying percentage of labeled positive images. **P+N** represents training with a percentage of reliable positives and the same percentage of negative labels, while **P+U** stands for training with a percentage of positives and all other images as unlabeled. The simoidal loss performs better than the weighted cross-entropy loss when the unlabeled positive samples dorminate. Training in the P+N settings is superior than training in the P+U settings with any percetage of positives labeled.

predict binary segmentation, determining whether a pixel belongs to an object or not. 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. The same hyperparameters for optimization were used as in Section 5.1. The trained models were evaluated with the test set of PASCAL VOC2011 segmentation dataset with binary segmentations.

As shown in Table 3, the sigmoid negative loss achieved the highest mean accuracy, approximately 0.07 better than training with the normal cross entropy loss. In contrast to the improvement of mean accuracy, mean IU for models trained with either weighted negative loss or sigmoid negative loss did not show significant improvement compared to the normal cross entropy loss. The increase in the mean accuracy was caused by an increase in foreground accuracy and a decrease in background accuracy. In other words, the sigmoid loss balances precision for recall, similarly as observed in training the CIFAR dataset. The decrease in precision, however, can interfere the improvement mean IU since the mean IU counts for both low false positive rate and low false negative rate.

We then applied the sigmoid negative loss to the pre-training phase of the experiment for inexhaustive segmentations in Section 5.1. We were able to recover the negative influence of inexhaustive segmentations on the fine-tuning performance of the pre-trained models. Note that we learned foreground/background segmentation instead of multi-class segmentation when applied the sigmoid negative loss because we discovered binarizing pre-training classes had little effect to feature transferability in this experimental setup.

Selective predictions made by models trained with the sigmoid negative loss and the cross entropy loss were presented in Figure 7. For these two example images, the model trained with the cross entropy loss failed to segment objects from images whereas sigmoid negative

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

Table 3: Foreground/background segmentation performance achieved on the test set of PASCAL VOC2011 segmentation dataset with the normal cross entropy loss, weighted negative loss, and sigmoid negative loss in the presence of inexhaustive segmentation. Mean accuracy is equivalent to the average recall over the two classes. Mean IU is the average intersection over union ratio (IU) over two classes and f.w. IU is the frequency weighted average of IU over the two classes. The sigmoid loss has a better mean accuracy than, and a similar mean IU as, the weighted loss. Both losses perform better than the normal cross entropy loss when 50% objects unsegmented. Experiments were repeated twice and the standard deviation of results is approximately 0.02.

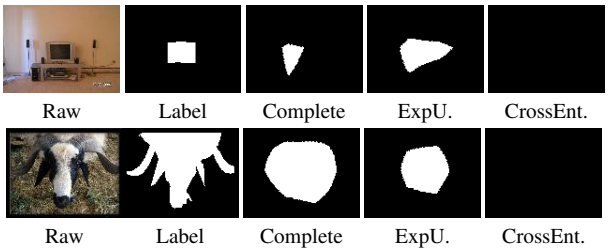


Figure 7: Selective predictions made by models trained with the sigmoid loss for negative class and the cross-entropy loss. This figure presented two selective images for which the model trained with the cross entropy loss failed to segment objects, and the one with sigmoid negative loss succeed.

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.

6 Discussion

We assumed incomplete segmentation, misclassification and false segmentation of objects had occurred independently to the exact shape and appearance of the objects when we synthesized the segmentation noises stochastically in our experiments. However, some categories can be more likely to be misclassified due to its ambiguity in shapes or appearances in practice, such as bear and teddy bear. Similarly, ambiguous objects can have a higher probability of being missing than easily recognizable objects. A real dataset with both clean and noisy segmentation would be valuable to evaluate the influence of noises on feature transferability further.

We proposed to binarize or categorize classes to produce accurate but imprecise labels when misclassification of segments dominates a small pre-training dataset. The result feature transferability depends on whether the inaccurate labels or the imprecise labels is more severe in the decrease of feature transferability. In our experiment, the limited amounts of training images, the low-resolution of images and the constraint capacity of the FCN-AlexNet model may explain that more precise labels cannot improve the fine-tuning performance for the pre-trained models. In general, a trade-off between precise but less ccurate labels and more accurate but less precise labels need to be made to train better transferable features, given a particular dataset.

We argued that not over-punishing confident, positive predictions for samples with negative labels can be beneficial by assuming the confident predictions are more likely to be mislabeled given a nonrandom model. However, there can also be disadvantageous for over-excusing losses for confident predictions. One typical problem is that a model made bad predictions with confidence will keep emphasizing its wrong predictions. Therefore, the threshold between punishing and excusing a certain model for make confident contrary predictions need to be tuned appropriately to achieve both high precision and high recall. Unfortunately, the sigmoid negative loss has no such hyperparameter to tune, which can be a direction to improve it in the future. For example, the Focal Loss[25] could be a parametrizable alternative of the sigmoid negative loss.

Besides, applying the sigmoid negative loss to segmentation assumes that the foreground-to-background mislabeling for each pixel occurred independently. This as-

sumption made it possible to implement sigmoid negative loss directly to classification for individual pixels. However, there is normally a spatial dependence for noises in pixel labels which may help improve the spatial consistency of predictions. Therefore, future works may be possible to make use of the spatial dependence of pixel label noises.

7 Conclusion

We investigate in this paper to pre-train transferable convolutional weights in the presence of inexhaustive segmentation, misclassification and false segmentation. With a simulated experiment, we discover that mislabeling of objects have negative influences on feature transferability. In particular, when misclassification of segments or incomplete segmentations is dominant in the pre-training data, fine-tuning the transferred weights is no better than random weights initialization. On the other hand, including false segmentations of meaningful objects for pre-training has little impact on the weights transferability for the pre-trained model. We present that binarizing classes as foreground and background can recover the decrease of feature transferability due to the misclassifications of object segments. 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 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 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] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

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

[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, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[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] 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.

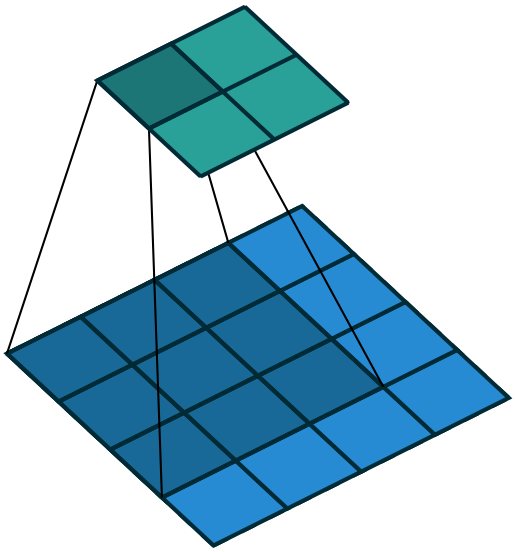


Figure 8: A basic convolution operation. A 3x3 convolutional filter convolves with a 3x3 window sliding over the image (bottom). The output of convolutions at each sliding position form a feature map (top). This figure was drawn by Dumoulin and Visin [6]

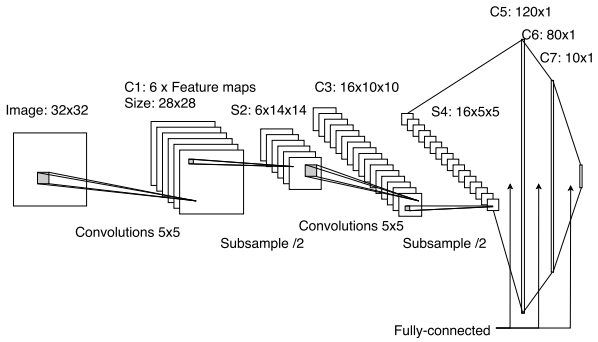


Figure 9: An example convolutional neural network, LeNet-5 [21]

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) [21], 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 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 [18], 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 shar-

ing 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 [26], 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. [28] 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. [28] 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.

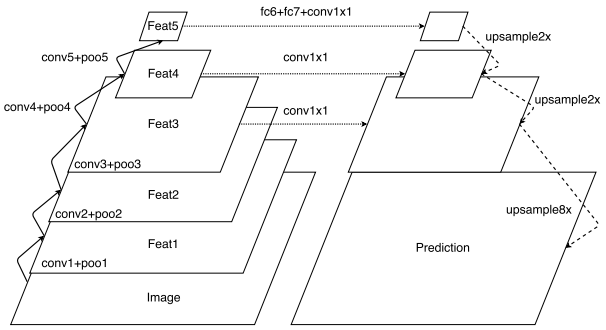


Figure 10: Fully convolutional network (FCN) by Long et al. (2015) [28]. Each

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:

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

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 $l+1$:

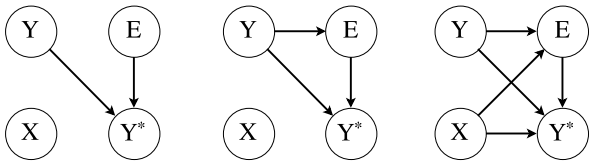
$$\delta^{(l)} = \left((\theta^{(l)})^T \delta^{(l+1)} \right) \bullet f'(z^{(l)})$$

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:

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



Noisy Completely at Random (NCAR) | Noisy at Random (NAR) | Noisy not at Random (NNAR)

Figure 11: Graphical models for three types of label noises. [11] The graphical models describe if the probability of the errorness and observed value of a label conditions on the true class and inputs. X is the random variable denotes the inputs; Y is stands for the variable for class; E is a binary variable determining errorness; Y^* represents a variable for the observed label;

C Label noises

Label noises can be categoried into three statistical models, noisy complete at random (NCAR), noisy at random (NAR) and noisy not at random (NNAR), based on whether label noises depends on classes and distribution of inputs. (Figure 11) For noisy completely at random models, the occurrence of an error E is independent of neither the true class Y nor the input X. The noisy at random model is similar to the noisy completely at random model except that the label noise is asymmetric, meaning it depends on classes. The noisy at random model model can be interpreted in terms of a transition (or confusion) matrix for K classes:

$$\gamma = \begin{bmatrix} P(Y^* = 1|Y = 1) & \dots & P(Y^* = K|Y = 1) \\ \vdots & \ddots & \vdots \\ P(Y^* = 1|Y = K) & \dots & P(Y^* = K|Y = K) \end{bmatrix}$$

where Y^* is a random variable for observed label and Y true label. The noisy not at random, on the other hand, represents that the label noise depends on both class and example.

In this paper, we assumed mislabeling of segmentations is noisy at random. TODO

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 CIFAR dataset classification.

D Supportive information

D.1 An 8-layer Convolutional neural network

D.2 Evaluation metrics

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

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

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

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

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

E Additional Results

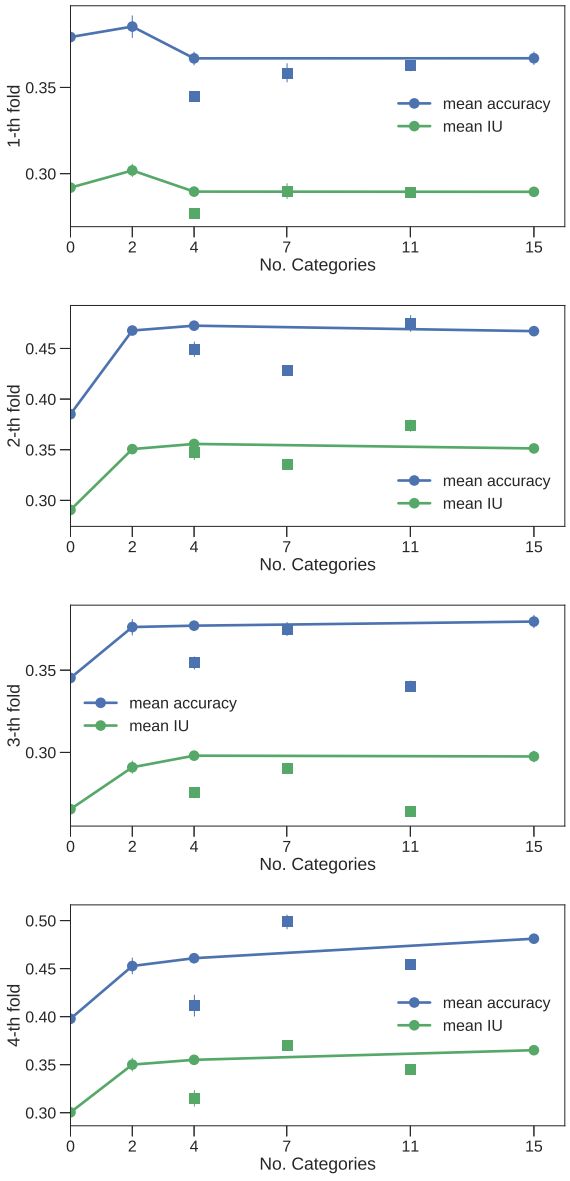


Figure 12: The influence of categorizing classes on test performance of fine-tuned models for each fold, addition to Figure 3.