1 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 16, 2017

Learn transferable features for semantic image segmentation in the presence of label noise

First Author
Institution1
Institution1 address
firstauthor@i1.org

Second Author
Institution2
First line of institution2 address
secondauthor@i2.org

August 16, 2017

Abstract

Given the mis-segmentation robustness of feature transferability discussed in section 6.1, we proposed to include mis-segmented objects to learn pre-trained model. Misclassification noises had also negative impact on weights transferability, but categorizing classes into categories or even binarizing can alleviate the randomness of labels while still achieve comparable fine-tuning performance. Lastly, inexhaustive segmentation can have significant influence on feature transferability and it can be compensate partially by modifying the loss function for the deep learning models. We proposed a particular class dependent loss in order to not over-punish confident contrary predictions. The proposed loss led to better pre-trainig performance as well as better fine-tuning performance with a synthesized dataset as discussed in section 6.2. The types of segmentation noises we concentrated on in this study are mis-segmentation, misclassification and inexhaustive segmentation.

2 Introduction

Why noisy labels? This paragraph should discuss the ubiquity of label noise and difficulties of collecting perfect annotations.
The recent success of deep neural networks benefits from the availability of large-scale supervised datasets such as [29, 5, 23, 19]. These datasets allow researchers to develop deep neural network models for object recognition[31], object detection[7], semantic image segmentation[20, 39] and other applications assuming the existence of perfect ground-truth segmentation. However, collecting well-annotated datasets on a large scale can be tremendously expensive and time-consuming in general. When annotators work on the tasks, it is natural for them to make mistakes as a result of lack of expertise, inherent ambiguity of tasks or unconscious bias. Enormous efforts were made in various manners to ensure the correctness of annotations as reported in, for example, [29, 19]. Saving efforts made for correctness will result in a noisy dataset but also potentially more annotated images. Trade-offs need to be made between the impact of by label noise and the gain of a larger dataset. For particular tasks, there exist freely available labels as alternatives to manual annotations. But these labels are often noisy owing to the way they were created. For example, one can use digital maps, like OpenStreetMap, to segment aerial images. These segmentations constructed from maps suffer from the incomplete annotation as well as registration problems.[22] Besides, Pl@ntNet[1], a crowdsourcing platform, provide millions of images of plants and corresponding labels which may or may not be correct. Strong supervision is therefore required to correct the mistakes, for example double-checking the annotations over and over again and ensembling opinions from multiple annotators. It is therefore sometimes inevitable for deep neural network models to accept the existence of noisy annotations.
Why feature's noise robustness? This paragraph should

[1]https://identify.plantnet-project.org/

discuss 1. CNNs arch: hierarchical feature extraction and task-specific layers 2. CNNs based models benefit from transferability of hierarchical feature
Convolutional neural networks (CNNs) based models often contain two principal components: a stack of convolutional layers to extract hierarchical features and a few task-specific layers to fit the specific learning objectives. [2] The convolutional layers were proved "transferable" not only to another dataset[36] but also to another application[7, 20]. This feature transferability allows reusing the convolutional features for tasks and applications different from which they were originally trained with in a transfer learning scenario, and it helps to achieve better performance when there are only limited number of training samples[20].
This paragraph should introduce the idea of studying the impact of annotation noise on feature transferability. In cases where only a clean but small dataset is available, an extra noisy but large dataset in a similar domain might be helpful as it can be used to pre-train the convolutional features of the CNNs model. Previous studies[32, 25] have reported a negative impact of label noises on classification performance, but not yet on convolutional feature transferability. In general, optimal classification performance on test set often indicates that the extracted features are also optimal, whereas suboptimal classification performance does not necessarily reflect the convolutional features are also suboptimal, especially concerning feature transferability. Feature transferability describes the *generality of features*, i.e., the category-independence of features. Low-level features were proved to be less dependent to categories and thus more transferable to new tasks than high-level features. [36] We experimented in Section 4 that how much label noises interfere the transferability of convolutional features.
Narrow the problem of discussion down to Segmentation In this paper, we considered three types noises that happen to semantic image segmentation: mis-segmentation, misclassification and inexhaustive segmentation, as described in details in Section 4.1. We chose to study segmentation errors because:

1. It is more difficult to correct noisy segmentation than to correct classification noises for object recognition;

2. Semantic segmentation can be treated as pixel-wise classification so that the existing noise-robust methods can be applied by assuming classifications for each pixels are made independently.

Explain why PU learning is relevant If we consider the inexhaustive segmenation issue only, i.e., the segmentation contains only a proportion of the target instances while leaving the rest unsegmented, the problem becomes similar to a so-called *positive and unlabelled learning* (PU learning) setup[17]. In the positive and unlabeled learning setup, the training dataset has two sets of examples: the *positive (P) set*, containing only positive examples, and the *unlabeled (U) set*, containing a mix of positive or negative examples. The segmented pixels in the presence of inexhaustive segmentation then form the P set and the unsegmented images construct the U set. Methods to learn with only positive examples and unlabeled examples become relevant because of the difficulty of distinguish object pixels and background pixels in the U set.
Table of contents
In the next section, we summarize related works in areas of transfer learning and learning with noisy labels for deep learning. In Section 4 we formulate the segmentation errors into three categories, mis-segmentation, misclassification and incomplete annotation, and test feature transferability against them separately. In Section 5 we connect training with inexhaustive segmentations to positive and unlabeled (PU) learning. We experiment with synthesized noisy dataset in Section 6.1 to study whether the mis-segmentation, misclassification and inexhaustive segmentation noises have impacts on learning transferable features. Section 6.2 contains experiments for methods to learn with positive and unlabled examples. Discussions are included in Section 7 and conclusions are summarized in Section 8.

[2]M: As a non-expert it is rather unclear to me how to identify or design generic layers and task-specific ones, or what it actually means or why it is important to make this distinction. In addition, I wonder to what extent end-to-end training actually turns generic layers into task-specific ones. J: I think the idea to make this distinction here is to highlight that hierarchical features are reusable for many different applications and tasks but task-specific ones are not as they may have different number of neurons due to the different number of classes, or have even completely different architectures because the learning objectives are different for different app/tasks.

3 Related work

Transfer Learning *transfer learning* We sometimes have a learning task in one domain of interest, but we only have sufficient training data in another domain which does not share a feature space with the domain of interest. Transfer learning arises in this scenario to transfer knowledge from one domain to anthoer and to improve the performance of learning by avoiding much expensive data-labeling efforts.[24] Weights of convolutional neural networks (CNNs) show outstanding transferability to another task. For example, weights trained on ImageNet images to perform image classification were shown successfully transfered to new categories and new learning problems[7, 20, 30]. Better performance were achieved for these tasks by using ImageNet pre-trained CNNs as initialization than training full model from scratch. Yosinski et al. discovered that feature transferability is negatively affected by the specialization of higher layer neurons and optimization difficulties caused by breaking co-adapted neurons. Their experiments showed that low-level features, which are less dependent to particular categories, are more transferable than high-level features. TODO:R Relations to our work. We studied if feature transferability is negatively affected by the presence of label noises.

Unsupervised pre-training Apart from supervised pre-training, one can also obtain pre-trained features in an unsupervised or a semi-supervised way. The most common method is to train a generative model with either *auto-encoder* variants or *deep beilief networks*. Vincent et al.[34] trained multiple levels of representation robust to the corrupted inputs with stacked denoising auto-encoders. Masci et al.[21] presented a stacked convolutional auto-encoder unsupervised pre-training for hierarchical feature extraction. Hinton et al.[11] proposed a greedy learning algorithm to train *deep belief nets* one layer at a time to train hierarchical features. Lee et al.[15] presented a *convolutional deep belief network*, to learn hierachical convolutional representations. A few studies[4, 3, 1] highlighted the advantage of unsupervised pre-training compared to the random initialization, connecting unsupervised pre-training to a norm of regularization and a method that help disentangle the sample variations. However, better random initialization strategies, for example, xavier initialization[8] and its variants, have shortened the gap between unsupervised pre-training and random initialization. Using unsupervised pre-training or not now becomes a tradeoff between the time and resources invested and the performance gain. Unsupervised deep representation learning is in general not comparable to supervised representation learning especially when large scale dataset is available.

Deep Learning with Noisy Labels A few studies[32, 25] investigated the impact of label noise on classification performance with convolutional neural networks assuming the labels were randomly transited from one to another given the probabilities fall in a transition matrix. They found a significant decrease in classification performance along with the increase of false label proportion when the total number of examples is fixed. They then proposed methods to handle this label noise at random (NAR)[6] situation by either introducing a linear noise layer on top of the output layer[32] or correcting the loss functions with an estimation of the noise transition matrix[25]. Xiao et al.[35] integrated a probabilistic graphic model to an end-to-end deep learning system to train predicting class labels, either correct or wrong, as well as to correct the wrong labels. Reed & Lee[26] proposed an empirical way of taking into account the *perceptual consistency* for large-scale object recognition and detection when incomplete and noisy labels exist by introducing a bootstrapping modification to the negative log-likelihood, in either a "Hard" or a "soft" favor.

Noise robustness In contrast to the works above, Rolnick et al.[28] argued that deep neural networks can learn robustly from the noisy dataset as long as an appropriate hyper parameters choice was made. They studied instead of replacing the correct labels with noisy labels but diluting correct labels with noisy labels to support their argument. They then concluded sufficiently large training set is of more importance than lower the level of noise. This work is closely related to our work in Section 4, except that we focus on the label noise robustness regarding the feature transferability instead of the classification performance. Additionally, most of these studies focus on the classification problems, whereas our work inclined more to the semantic segmentation problem.

Positive and Unlabeled Learning The previous studies about PU learning mainly focus on the binary classification for linear-separable problems[2, 16], whereas we showed in Section 5 that it is possible to train deep neural networks for multiple classes with only "positive" and unlabeled examples.

4 Noise robustness for convolutional feature transferability

Feature hierarchy, feature generality and feature transferability

Convolutional neural networks for processing images are believed to extract a rich hierarchy of visual features that vary from simple to complex and from local to global.[7] Neurons in the lower hierarchy capture simple patterns such as edges, textures or material properties within small receptive fields, whereas the high-level features capture conceptual information such as people and text in larger receptive fields. It is natural that simple patterns can be found for many categories whereas conceptual features are often specific to particular categories. By training a CNN with half of the ImageNet categories and transfer features to the other half, Yosinski et al.[36] found transferability of features decreases from the bottom layers to the top layers, indicating that features change gradually from category independent to class dependent. Additionally, filters from the first convolutional layer are often observed as Gabor filters or color blobs regardless the training dataset and target categories.[38, 15, 13, 30] Correspondingly, low-level filters, especially ones from the first two layers, were proved well-transferable to dissimilar categories, as reported in [36] from natural objects to man-made objects. These category-independent convolutional filters are denoted *general* in the sense that they are generally usable for many classes.

Given the evidence that some convolutional filters can be applied regardless the exact categories they were trained, we wondered if they can also be learned in a more general way than using the exact labels of specific classes. For example, labels assigned examples from categories that are visually similar can be inconsistent in practice if not additional human efforts made for correctness. These noises in labels may have a significant negative influence to the top layers which are sensitive to the given labels but not necessarily to the bottom layers which have already shown class independence to some extent. If we train models in a transfer learning scenario where we only care about the transferability of a pre-trained model rather than performance achieved on the pre-training task, a label noise robustness of weights transferability can be beneficial to save additional human efforts for label correction. We studied the influence of three types of segmentation noises on weights transferability in Section 6.1. The research problem is formulated in Section 4.1 and the segmentation noises of interest are described in Section 4.2.

4.1 Problem Formulation

Semantic Segmentation A deep learning model for semantic segmentation normally consists of two main components: a CNN feature extractor $G : R^{h\times w\times c} \to R^d$ and a pixel-by-pixel classifier $H : R^d \to R^{h\times w\times k}$, where h, w are image height and weight respectively; c is the number of image channels and k is the number of classes; d is the total number of extracted features. Together they form a segmentation model $F : R^{h\times w\times c} \to R^{h\times w\times k}$ to predict class probabilities for each of the pixels in a given image x:

$$P(y|x;\theta) = F(x) = H(G(x))$$

where θ denotes the model weights.

Feature Transferability In practice, feature extractor G can be transferred from a pre-trained model. The transferred feature extractor G' can be used directly in F, or used as an initialization of G and fine-tune with the dataset of interest. The pre-trained model is normally an ImageNet model trained by the object recognition task. Alternatively, one can also obtain a pre-trained model with semantic segmentation task if there exists a segmentation dataset in a more similar domain than ImageNet. The transferability of a pre-trained feature extractor can be measured by performance achieved by F with transferred G' on the test set.

Supposing a pre-training segmentation dataset have both noisy labels $\tilde{y}$ and true labels y available, the influence of label noises can be studied by comparing the transferability of feature extractors learned with y and with $\tilde{y}$. It is difficult to find such a dataset with both clean and noisy labels, so we tried to synthesize segmentation noises with well-annotated labels.

4.2 Label noise synthesization

How noises synthesized For segmentation problems, each pixel (or voxel for 3D segmentation) of an training image has a label assigned to one of the pre-defined categories. Supposing there are K pre-defined categories, the label of pixel ij

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

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

A straightforward way to synthesize noisy labels is to corrupt stochastically true labels for each pixel with a corruption model. The corruption model describes the probability of an observed label conditioning on the the true label y_{ij}, the image x and the label errorness e_ij:

$$p(\tilde{y_{ij}}|x, y_{ij}, e_{ij})$$

where the binary error occurence e_{ij} depends on the inputs x and true labels y. Given a corruption model and true labels, one can stochastically synthesized the corresponding noisy labels.

Clarity for noises considered. In our works, we considered three types of noises for a semantic segmentation problem, *mis-segmentation*, *misclassification* and *inexhaustive segmentation*, and modified the above corruption model for each type of noise accordingly. Note that all these label errors apply to the whole segment instead of to individual pixels. That is pixels for the same segments will have the same true labels and observed labels. Therefore, the above corruption model hides the spatial dependence of e_{ij} from expressions. We exluded segmenting errors such as inprecise boundaries, oversegmenting or undersegmenting the objects from study because they are a bit more complex to synthesize than the preceding classification errors.

Mis-segmentation Mis-segmentation denotes the wrongly segmented objects for categories that are semantically meaningful but are not predefined. For example, a toy dog can be misannotated as a dog, assuming that "dog" is predefined and "toy dog" is not. In the presence of mis-segmentation, pixel labels of segment S transit from 0 to k with probability

$$p(\tilde{y_{ij}} = k|x, y_{ij} = 0), ij \in S$$

The dependence of observed pixel labels $\tilde{y_{ij}}$ on the original image x interpret the premise that mis-segmentation would only happen to semantically meaningful segments in an image. It is natural to include this premise because semantically meaningless partitions of an image are less likely to be segmented by an annotator. However, it is difficult to estimate the above proability in practice because it is conditioning on the semantic meaning of x. Therefore, we synthesized mis-segmentation errors by selecting part of the categories as non-target categories so that instances of these categories should have zero labels for correct segmentations. We can then misannotate these non-target instances stochastically with a simplified probability $p(\tilde{y_{ij}} = k|y_{ij} = 0) = p_k$ without interpreting semantic meaning of x in probability. Note that p_k sums up to 1 for all classes $\sum_0^K p_k = 1$.

Misclassification Different from mis-segmentation, misclassification error means labels were misclassified between pre-defined categories. For example, cats may be misclassified as dogs occasionally if both "cat" and "dog" are target classes. In the presence of misclassification, pixel labels of segment S are transited from k to j stochastically with probability:

$$p(\tilde{y_{ij}} = j|y_{ij} = k) = p_{jk}, ij \in S, k, j \in [1, K]$$

where $\sum_{j=1}^K p_{jk} = 1$. We assumed misclassification error is independent of the exact shape and appearance of the objects, i.e.information from x. This model is often called *noisy at random* [6]. This assumption does not hold in every cases of practice, for example, some instances can be more likely to be misclassifiied due to its ambiguity in shapes or apperances. But the difficulty of modeling the depence of x leads to simply assuming an input indepence. Given the class transition probabilities, one can easily synthesize noisy annotations including misclassification errors given a well-annotated segmentation dataset.

Inexhaustive segmentation Inexhaustive segmentation denotes that there exists unsegmented instances for predefined categories. Pixels of an unsegmented object S have labels flipped from k to 0 with probability:

$$p(\tilde{y_{ij}} = 0 | y_{ij} = k) = q_k, ij \in S, k \in [1, K]$$

In words, an instance of category k is left unsegmented stochastically with probability q_k. The probability of correctly segmented in annotations is then:

$$p(\tilde{y_{ij}} = k | y_{ij} = k) = 1 - q_k, ij \in S, k \in [1, K]$$

Again we assumed inexhaustive segmentations to be noisy at random (NAR).

5 Positive and Unlabeled Learning

This part should explain the necessarity of Positive and Unlabeled Learning setup Experiments in Section 6.1 indicates that inexhaustive segmentation can have significant negative influences on feature transferability. Besides, including mis-segmented objects for training can aggravate the inexhaustive segmentation problem. For example, the existence a mis-segmented toy dog does not mean that every toy dogs are mis-segmented. The other unsegmented toy dogs then become a source of inexhaustive segmentation and lead to worse fine-tuning performance as we discovered in Section 6. Method to compensate inexhaustive segmentation is therefore necessary to train better transferable representation.

Learning with inexhaustive segmentation fits a PU learning setup *This part should formulate the PU learning probleml and discuss why PU Learning instead of semi-supervised learning is the way to go.*
To simplify the demonstration, let us consider a binary segmentation problem where we have *positive* and *negative* pixels denoting pixels corresponding to target and non-target segments. The goal of compensating incompleteness is to learn a model that predicts as many positive pixels as possible while keeping the false positive rate low. A pixel with a negative label can be either a truly negative pixel or a wrongly unsegmented positive pixel. In other words, there is no explicitly labeled negative examples in the training set. This naturally fit in a Positive and Unlabeled Learning (PU Learning) setup where the training dataset consists of only a set of positive examples (P set) and a set of unlabeled examples (U set), meaning that there are no reliable negative examples available. The traditional semi-supervised learning techniques are not applicable in such a setup as a result of the absence of negative training samples.

Class-weighted Logistic Loss *This part should discuss the difficulty of applying traditional PU learning methods to deep learning and then introduce the idea of re-weigh positive and negative classes.* Traditional PU learning methods often follow a two-step strategy: first identifying a set of reliable negative samples (RN set) from U set and then iteratively build classifiers, either nave Bayesian (NB) or supported vector machine (SVM), with RN and P sets and update the RN set with expectation-maximization (EM) algorithm. However, this strategy requires training multiple classification models. It is unrealistic to train iteratively a sequence of deep learning models because it would take significantly longer time to train neural networks than to train a nave Bayesian (NB) model or supported vector machine (SVM).
Our original goal was to achieve high precision as well as high recall regardless the existence of false negative labels. A straightforward way to achieve this goal is to simply reweigh the positive and negative examples, namely let the positive and negative examples have different rates of contribution to the total loss. Suppose logistic loss is used, the corresponding losses for positive and negative samples are:

$$\begin{aligned} l_{\tilde{y_i}=+1} &= -\log p(y_i = +1 | x_i) \\ l_{\tilde{y_i}=-1} &= -q \log p(y_i = -1 | x_i), 0 < q < 1 \end{aligned}$$

where $p(y_i|x_i) = \sigma(f(x))$ denotes the probablistic output of the model $f(\cdot)$ for the i-th example. Emprically, the choice of q can be made based on the most highest precision and recall achieved on validation set. Alternatively, one can also roughly assign $q = p(y = -1 | \tilde{y} = -1)$. This turns out to be part of the backward corrected loss proposed in [25]:

$$\begin{aligned} l_{\tilde{y_i}=-1} = &-p(y_i = -1 | \tilde{y_i} = -1) \log p(y_i = -1 | x_i) \\ &-p(y_i = +1 | \tilde{y_i} = -1) \log p(y_i = +1 | x_i) \end{aligned}$$

with $p(y_i = -1|\tilde{y_i} = -1) = q$ and $p(y_i = +1|\tilde{y_i} = -1) = 1 - q$.

Confidence of contrary prediction and probability of false negative label *The idea of alleviating punishment for confident predictions.* Losses above assume that the true label of an observed negative label is independent of the inputs x. However, the true label y is often dependent on both x and $\tilde{y}$ in practice. For instance, in segmentation problems, whether a pixel is left unsegmented correctly or not can be determined by the semantical meaning of this pixel and pixels around. It is nevertheless difficult to estimate $p(y|x, \tilde{y})$ directly due to the difficulty of exploring the joint distribution of x and $\tilde{y}$. Alternatively, one can use the probabilistic prediction $p(y|x; \theta)$ a classification model to provide extra information about the inputs. The predicted probabilities indicate how confident the current model is for the predictions made. Given a good enough model, more confident positive predictions for negatively labeled examples can have a higher probability of being wrongly annotated than the less confident ones. [3] With a random model by which predictions are made randomly, the high and low confidences do not convey any information about the underlying inputs distribution. By contrast, confident predictions made by a trained model indicate that the corresponding samples are possibly close to some previous training samples with positive labels in the feature space. The natural trend of similar examples having the same labels supports a higher probability of being annotated wrongly for confident predictions which have opposite labels. This thought leads to the idea of not punishing confident contrary predictions more than unconfident ones as the traditional logistic/softmax loss will do.

Exponential Loss for unlabeled examples *This section should mention the class dependent losses and introduce ExponentialUnlabeledLoss* In a PU learning setup, the positive (P) set contains only reliable positive labels, whereas the unlabeled (U) set can be considered as noisy negative labels. The problem then converts to training with clean positive examples and noisy negative examples. We used a class-dependent loss to compensate the noisy negative labels while still making full use of the clean positive labels. The loss was made of a normal logistic loss for positive examples and an exponential loss [33] for examples with negative labels:

$$l_{\tilde{y}_i=+1} = -\log p(y_i = +1|x_i)$$
$$l_{\tilde{y}_i=-1} = 1 - p(y_i = -1|x_i)$$

Figure 1 shows the weighted logistic loss with $q = 0.5$ and the exponential unlabeled loss respectively and their derivatives with respect to logits by varing the logit from negative to positive. The main feature of exponential loss is its relatively small changes in the region of confident positive for negatively labeled examples, compared to the logistic loss and class weighted logistic loss. As a consequence of this feature, the corresponding derivative decreases to zero as the prediction increases in the positive direction. This feature interprets the idea of not punishing positive prediction with confidence for negatively labeled samples. Another modified loss function, namely the Focal Loss[18], instantiates the same idea with a similiar form of expression.

This section discuss the loss and derivative difference between the losses. Figure 2 shows the loss and derivate difference between logistic loss, class-weighted logistics loss and exponential negative (unlabeled) loss with a two-dimensional example. It demonstrates that, for exponential unlabeled loss, unlabeled positive examples farther from the decision boundary do not have larger loss contributions as ones closer to but still distant from the decision boundary. Confident positive predictions, shown as examples located on the positive side of the decision boundary and far from it, has little effect for updating model weights. The consequence of this effect is that positive examples push the decision boundary away from the positive cluster while negative examples closed to the decision boundary instead of those away from the decision boundary pull the decision boundary towards the positive cluster. This characterisitc of exponential loss lead to a selectively counting weights update contributions of negative samples than simply lower the overall estimation for all negative samples while optimization. The exponential loss was introduced in [33] to get rid of the effect of outliers. In our case, the negative examples given confident predictions by classifier can be considered as outliers.

[3] J: This argument needs more detailed explanation.

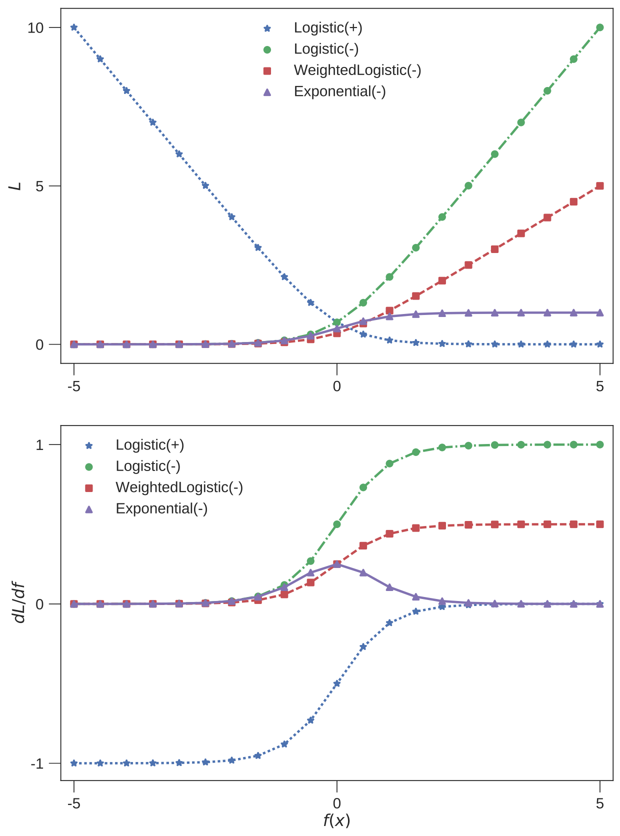

Figure 1: The Logistic Loss, Weighted Logistic Loss, Exponential Loss and their derivatives with respect to logits.

Minimum entropy regularization and bootstrapping objective An alternative way of encouraging confident positive predictions is to introduce minimum entropy regularization[9]. Reed et al.[26] proposed an emprical modification to softmax loss, a.k.a. the *bootstrapping loss*, which can be used to learn with unlabeled examples:

$$l_{\tilde{y}_i=-1} = -\beta \log p(y_i = -1|x_i) - (1-\beta) \log p(y_i = \hat{y}_i|x_i)$$

where $\hat{y} = argmax_{j\in\{-1,+1\}} p(y_i = j|x_i)$ is the model prediction and $0 < \beta < 1$. The first term of the objective is a weighted logistic loss and the second term can be considered as a variation of minimum entropy regularization:

$$\begin{aligned} H &= -\sum_{j\in\{-1,+1\}} p(y_i = j|x_i) \log p(y_i = j|x_i) \\ &\sim -\sum_{j\in\{-1,+1\}} \delta(y_i - \hat{y}_i) \log p(y_i = j|x_i) \end{aligned}$$

which intuitively encourages the model to make confident predictions[9].

Multiclass PU learning *This part should describe how to extend the losses for multiclass scenarios and the difficulty of having multiple positive classes.* So far, we have been discussing modifications to the logistic loss for binary classification and segmentation. Similiar modifications can be applied to softmax loss as well. In a multiclass positive and unlabeled learning setup, $K > 1$ positive classes are supposed to be distinguished from one negative class, whereas only part of the positive examples are labeled, and the others are not. Softmax loss can be still used for positive classes because positive labels are reliable as long as they are given. By contrast, the negatively labeled samples are a mix of samples assigned with true negative labels and false positive labels. The weighted unlabeled logistic, exponential unlabeled loss and bootstrapping can be used for negative by simply replacing logistic function with softmax function as the activation function for logits.

From classification to segmentation *This paragraph should highlight the problem of spatial independence assumption.* Classification for each pixel is made independently when segmentation task is considered as per-pixel classification. The objective modifications to alleviate the negative impact of unlabeled positive samples can be applied to classification for each pixel by assuming the probability of missing positive label for a pixel is independent of its neighbor pixels, which may not hold in practice. For example, we stochastically flipped pixel labels for the whole segment, instead of individual pixel label, to synthesize incomplete segmentation errors in Section 4, Even with annotation errors like under-segmented objects, the probability for a pixel to be unsegmented depends on its neighbor pixels. We made a strong assumption of spatial independence for false negative labels for inexhaustive segmentation noises. Future works maybe required to

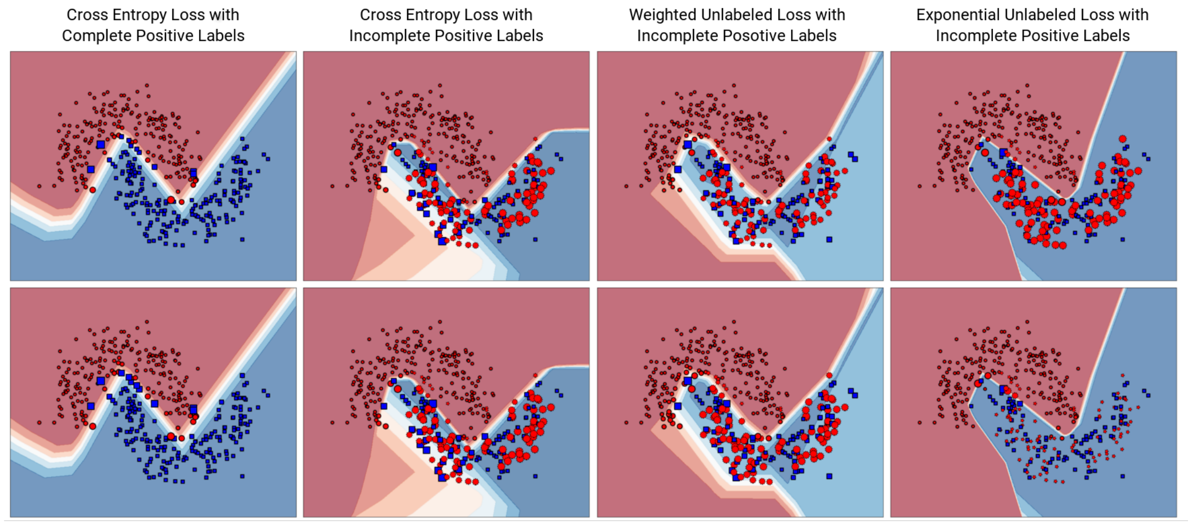

Figure 2: 2D moons dataset with non-linear separable decision boudary. Four hundreds samples per class were drawn randomly from two interleaving half circles with noises added with a minor standard deviation. A **red circle** indicates an example labelled as positive whilst a **blue square** indicates the example has a negative label. The **leftmost** figures have complete positive labels, meaning the positive and negative labels are all correct, whereas, in **the other figures** only half of the positives were correctly labelled and the rest were mixed with the negative samples. 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 **size of the markers** in the top row denotes the per-class normalized training losses and the **size of the markers** in the bottom row the per-class normalized derivatives w.r.t the output of the last layer for the trained Multilayer Perceptron (MLP) with the different losses.

relax this assumption and make use of the spatial dependence of label noises to achieve higher mean intersection over union (mean IU) not only higher accuracy.

Implementation details *This paragraph should explain fade-in was introduced to avoid all-positive inital prediction;* The exponential loss for negative (unlabeled) examples saturates for very positive outputs, meaning that the confident positive prediction has little contribution to the total loss. This can introduce problems at the beginning of the training procedure when the confident predictions are likely to be made at random. Additionally, optimization could reach the plateau where the model made all positive predictions with high confidence. Therefore, we introduced the exponential loss after training with the normal logistic/softmax loss for a few epochs. We applied a similar "fade-in" mechanism to the bootstrapping objective as well because it also requires a non-random model for sufficiently trustworthy prediction $\hat{y}$.

This part should explain the influence of the imbalanced problem and how to overcome. Another problem encountered in the PU learning setup is the class imbalance introduced by negatively labeled positive samples. Even 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 addtion to this frequency balancing class weight.

6 Experiments

6.1 Synthesized mis-segmentation, misclassification and inexhaustive segmentations

Experiment setup In order to investigate the influence of mis-segmentation, misclassification and inexhaustive segmentation on feature transferability respectively, we set up experiments with a perfectly annotated dataset, the PASCAL VOC2011 dataset[5]. Fifteen out of twenty categories 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[20] for segmentation in the precense or absence of synthesized segmentation errors. The fine-tuning dataset was used to fine-tune the convolutional weights from the pre-trained FCN-AlexNet models. Fine-tuned models were then evaluated by mean intersection over union ratio (mean IU) achieved on the test set of fine-tuning dataset, referring to as the *fine-tuning performance*. Performance improvement of fine-tuning models compared to an randomly initialized model indicates the transferability of pre-trained weights.

Experiment details To avoid 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. Each fold was studied separately, and the exact partitions of each fold is listed in Table 1. The training dataset was enriched with extra segmentations by Hariharan et al.[10] 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 dataset and evaluation. In order to accelerate the training process, we subsampled the original images by four times. Fully Convolutional Networks with AlexNet was used for experiments because its relatively small capacity and thus short training time. The existence of an ImageNet model for AlexNet can be beneficial to set a performance reference. Only convolutional filters of AlexNet were transfered from the pre-training phase to fine-tuning phase because the transferability of convolutional weights were the focus of this work. The other layers were random initialized with Xavier Initialization. The ImageNet model and completely random weight initialization were considered as the upper bound and lower bound, respectively, for various pre-trained weights summarized in Table 1. The default hyperparameters of FCN-AlexNet in [20] were kept unchanged. Training 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.

Feature Transferability robustness to segmentation noises *What Table 1 tell us. How annotation errors were synthesized; How synthesizations are different from* *reality; Transferability of noisy models compared to clean models.* Mis-segmentation, misclassification and inexhaustive segmentation were synthesized separately with stochastical corruptions to the well-annotated pre-training dataset followed the descriptions in Section 4.1.

Mis-segmentation: little effect on weights transferability To synthesize mis-segmentation noises, 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 4.1. In the presence of mis-segmentation noises, instances from non-target categories can be misannotated as the target category with probability of $p_1 = 1$ and $p_1 = 0.5$ respectively. The two choices of probability led to two different pre-training sets and thus two different pre-trained models, naming the AllMisSegmented model and the HalfMisSegmented model, in Table 1 respectively. The noise-free counterpart of mis-segmentation is an dataset containing segmentations of the selected target category only whilt the other 14 categories remained unsegmented. Model trained with this noise-free dataset was denoted as NoMisSegmented in Table 1.

Table 1 shows that all three models achieved better fine-tuning performances than random initialization. The dataset mis-segmented all non-target instances produced a model with even slightly better fine-tuning performance than the dataset segmented only the target category. However, it is less likely to happen in practice that annotators will mis-segment every single non-target instance in the dataset. Mis-segmentations often occurr in annotations occasionally. Therefore, we also trained models with a training set containing half of the non-target objects to test if inexhaustively mis-segmenting non-target objects decreased the fine-tuning performance. The results show that the HalfMisSegmented model had a slightly worse fine-tuning performance than the AllMisSegmented model but was still comparable to the NoMisSegmented model. Based on these observations, we concluded that mis-segmenting semantically meaningful objects could have little impact when they are used for pre-training transferable weights.

Misclassification: negatively affect weights transferability Misclassification errors were also synthesized from the well-annotated pre-training dataset as described in Section 4.1. The noisy dataset containing labels for target segments stochastically transited to a random class with probability $p_{jk} = \frac{1}{20}$. The resulted trained model was denoted as the AllRandomLabels model in Table 1. Similiarly, if a random half of the segmented objects were assigned random labels, the resulting pre-trained model is called the HalfRandomLabels model. The noise-free counterpart of these two noisy models was the model trained with true labels, denoting as the TrueLabels model.

Compared to the TrueLabels model, the noisy models trained with samples containing random labels, no matter if all labels were random or if only half of the labels were random, led to worse fine-tuning performances. Fine-tuned performances of the AllRandomLabels model and the HalfRandomLabels model were no better than randomly initializing model weights, indicating poor weights transferabilities of a trained model in the presence of random labels to segmentations. In other words, misclassification noises in segmentation can impact the transferability of convolutional weights negatively.

Inexaustive Segmentation: negatively affect weights transferability Inexhaustive segmentations in the training dataset were synthesized by randomly converting labels of segmented objects to 0 with probability $q_k = 0.5$. Similiar as misclassification errors, inexhaustive segmentation can have negative impact on weights transferability. Pre-trained model trained from a dataset with 50% percentage of the instances unsegmented produced a fine-tuned model with an average mean IU 0.04 worse than the model pre-trained with true labels and it was almost the same as training a model with random weight initialization.

Categorizing classes for pre-training *This paragraph should discuss the potential benefit of categorizing classes. It had equivalent fine-tuning performance as using true labels; It achieved better performance than training to segment the exact classes but with misclassification labels.* Table 1 also showed that the fine-tuning performance of weights (AllMisSegmented) trained with binarized labels was better than weights (HalfRandomLabels) pre-trained with random class labels. This observation can be relevant for learning transferable weights in the presence of misclassification because one can train binary segmentation if the pre-training dataset is dominated by misclassification errors. Besides, weights pre-

Initial Representation	mean IU (aeroplane, bicycle, bird, boat, bottle)	mean IU (bus, car, cat, chair, cow)	mean IU (dining table, dog, horse, motorbike, person)	mean IU (potted plant, sheep, sofa, train, TV)	avg mean IU and avg std.
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
NoMisSegmented	0.26 ± 0.01	0.37 ± 0.03	0.27 ± 0.01	0.33 ± 0.04	0.31 ± 0.02
AllMisSegmented	0.30 ± 0.02	0.35 ± 0.01	0.29 ± 0.02	0.35 ± 0.03	0.32 ± 0.02
HalfMisSegmented	0.27 ± 0.01	0.34 ± 0.01	0.30 ± 0.01	0.32 ± 0.01	0.31 ± 0.01
ExponentialU.	0.31 ± 0.00	0.37 ± 0.00	0.33 ± 0.00	0.35 ± 0.00	$0.xx \pm 0.00$
TrueLabels	0.29 ± 0.01	0.36 ± 0.01	0.29 ± 0.01	0.37 ± 0.01	0.33 ± 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
InexaustiveSegmented	0.26 ± 0.01	0.30 ± 0.3	0.28 ± 0.03	0.32 ± 0.02	0.29 ± 0.02

Table 1: Performances of fine-tuned FCN-AlexNet models with different representation initializations. **ImageNetModel** represents the pre-trained ImageNet model; **RandomWeights** indicates that the weights were randomly initialized; All the other weights were first trained with the pre-training dataset in the presence or the absence of different types of label noises. Each experiment was repeated three times, the mean and the standard deviation were computed over the last five snapshots for all repetitions.

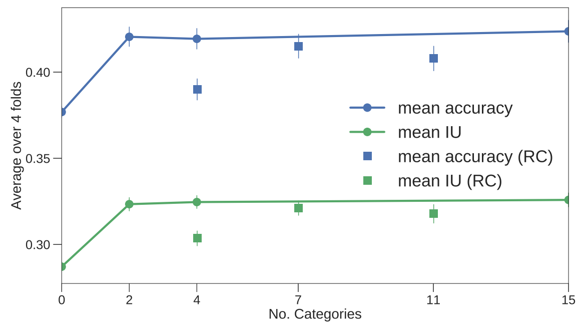

Figure 3: Test performance for fine-tuned models initialized with weights pre-trained with categorized 15 classes. Isolated error bars located aside from the lines denote random categorizations (RC) of the 15 classes. The displayed mean IU/mean accuracies and standard deviations were averaged over four folds listed in Table 1. Experiments were repeated three times.

trained by binary segmentation, segmenting object pixels from the background, can have a comparable fine-tuning performance to weights pre-trained by multi-class segmentation. In our experiment, the number of samples for each class in the pre-training dataset was limited. Most of the classes had only around one hundred images, and the dining table had only 20 images. The limited number of class samples can increase the difficulty to segment individual classes and may explain why binarized labels could achieve comparable fine-tuning performance to true labels. We then studied the influence of varying number of categories for pre-training. We categorized the fifteen classes in the pre-training set into person, animal, vehicle, indoor according to [5] and trained , shown as the error bars on lines in figure 3 at categories=4. The fifteen classes were also randomly categorized into 4, 7, 11 categories and shown as separate error bars in figure 3 at categories=4, 7, 11 respectively. Figure 3 shows that categorizing classes into categories had little effect to the fine-tuning performance of trained weights. Even categorizing classes into random categories without explicit meaning could pre-train weights better than random initialization. Binarizing or categorizing classes into higher hierarchy can be beneficial when the main learning objective is to train transferable convolutional weights, and the training dataset is corrupted by noisy labels.

6.2 PU Learning for classification and for segmentation

In order to compare exponential unlabeled loss with class weighted logistic/softmax loss, we synthesized positive and unlabeled learning setups for classification and segmentation respectively.

In the classification setup, we combined the CIFAR10 dataset and CIFAR100 dataset, using CIFAR10 as the positive (P) set and CIFAR100 as the negative (N) set. The learning objective is to classify images into eleven classes: ten positive classes from CIFAR10 and a negative class for images from CIFAR100. Note that there is no category overlap between CIFAR10 dataset and CIFAR100 dataset. To synthesize a positive and unlabeled (PU) learning setup, we selected only part of positive images from CIFAR10 to be correctly labeled and the rest of the CIFAR10 images were mixed with CIFAR100 images, forming the unlabeled (U) set. Models were then trained with the labeled part of P set and U set. An eight layer VGG net was used together with different choices of losses. The architecture of this VGG8 model can be found in Appendix TODO:Jihong. Each model was trained from scratch with Adam optimizer base and learning rate 0.0001. Model performances were evaluated on

Annotation	Loss	acc.	mean prec.	mean rec.	mean F_1
Complete	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	0.39	0.50
50%P+U	WeightedU.	0.78	0.75	0.75	0.76
50%P+U	ExponentialU.	**0.81**	**0.85**±0.03	0.72±0.03	0.77
50%P+U	BootstrapHard	0.80	0.76	**0.81**	**0.78**

Table 2: Accuracy, mean precision, mean recall and mean f1-score on test set of the CIFAR dataset with true labels.The complete dataset contains images from CIFAR10 as the **positive** (P) set and images from CIFAR110 as the **negative** (N) set. The unlabeled positive examples from P set construct the **unlabeled** (U) set, together with N set. Precision and recall were averaged over ten positive classes. 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.

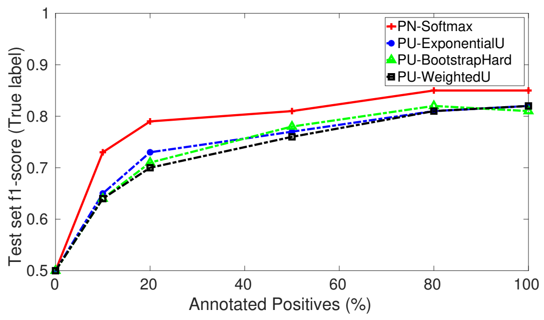

Figure 4: Varying percentage of labeled positive images. **P+N** represents training with percetage of images with reliable positive and negative labels and **P+U** stands for training with the positive (P) and unlabeled (U) sets.

a separate test set of combined CIFAR10 and CIFAR100 with true labels.

Table 2 summarizes the test precisions and recalls for training with different losses in the PU setup, compared with training with complete positive labels. With a training set containing 50% labeled CIFAR10 images and the rest unlabeled, the normal cross-entropy loss led 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, we were able to balance precision and recall and improve the resulting f1-score. Compared to the negative weighted loss, exponential loss and (hard) bootstrapping loss were able to achieve slightly better f1-scores. TODO: explain Training f-1 0.83 vs 0.81

As a reference for the performances, we trained a classifier with 50% of the positive samples and the same percentage of true negative samples. We refered this setup as positive and negative (PN) setup. The total number of training sample in PN setup is smaller because the rest unlabeled positive and negative samples were excluded from training. In Figure 4 we varied the percentage of labeled positive images, and compared the three different losses in the PU setup with a normal cross-entropy loss in the PN setup. In any of the labeled percentages for positive images, training with positive and negative examples can achieve higher f1-scores than any of the models trained with the same amounts of positive images and unlabeled images. The performance difference between learning with PN and learning with PU increases as the number of labeled positive images decreases. This result was expected because PN setup delivers extra information about which images in the unlabeled set are negative. The PU setup is therfore only relevant when it is difficult to annotate negative examples from the unlabeled data. And segmentation problem in the presence of inexhaustive segmentation can be such an example.

In the segmentation setup, we used again the PASCAL VOC2011 dataset with extra segmentation[10]. We synthesized inexhaustive segmentations the same way as described in Section 6.1. The same AlexNet-FCN model were trained together with the different loss functions for class 0 to predict binary segmentation, determining whether a pixel is correspondent to an object or not. Only single-object images were used for training and testing in order 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 6.1. The trained models were evaluated with the test set of PASACAL VOC2011 segmentation dataset with binary segmentations.

As shown in Table 3, the exponential unlabeled loss achieved the highest mean accuracy and a slightly lower overall accuracy. In contrast to the improvement of mean accuracy, mean IU for models trained with either weighted unlabeled loss or exponential unlabeled loss did not show significant improvement to the normal cross entropy loss.

Annotation	Loss	overall acc.	mean acc.	f.w. IU	mean IU
Complete	CrossEnt.U	0.90	0.85	0.82	0.75
50%Unseg.	CrossEnt.U	0.85	0.68	0.73	0.60
50%Unseg.	WeightedU	0.84	0.71	0.73	**0.62**
50%Unseg.	ExponentialU	0.83	**0.75**	0.72	**0.62**

Table 3: Best binary segmentation performance achieved on the test set of PASCAL VOC2011 segmentation dataset in the presence of inexhaustive segmentation. Class weight 0.7:1.75 was used to balance the sample frequency difference of the two classes and negative loss were further weighted by a factor of 0.5 for weighted unlabeled loss. Mean accuracy is equivalent to mean recall over 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. Experiments were repeated twice and standard deviations were approximately 0.01.

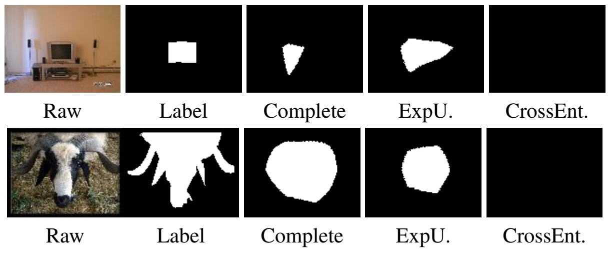

Figure 5: Selective predictions for models in Table 3.

Selective predictions for models trained with exponential unlabeled (ExpU.) loss and normal cross entropy (CrossEnt.) loss were presented in Figure 5. For these two example images, the model trained with cross entropy loss failed to segment objects from images whereas exponential unlabeled loss segmented on the position of the objects though with coarse outlines. The third column shows predictions given by model trained with complete training segmentation, and it did not give more accurate outlines. The coarse results were mainly due to the limited compacity of AlexNet model.

Exponential loss help improve fine-tuning performance Additionally,

7 Discussion

Why mis-segmentation can have little negative effect?

Why binarizing/catergoring classes have little negative effect?

Upsides and downsides of exponential loss Upside:

- not over-punish confident positive prediction for negatively labeled examples
- easy to implement with Pixel independent noise assumption

Downside:

- Non-parameterizable
- Optimization diffilculty introduced as a result of non-convex objective

Future works

8 Conclution

- Feature transferability is robust to mis-segmentation but not to misclassification and inexhaustive segmentation.
- Misclassification noises can be alleviated by binarizing/categorizing classes.
- Inexhausitive segmentation can be translated as learning with only positive and unlabeled examples.
- We proposed a class dependent loss to not over-punish confident positive predictions in the presence of noisy negative labels, and it showed slightly better results than class-weighted loss.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

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

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[38] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[39] 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.

A Convolutional Networks for Semantic Segmentation

A.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) for image processing is often comprised of stacked convolutional layers with interlacing subsampling layers for feature extraction, followed by a few other layers for either classification or regression, depending on the task-specific objectives. For instance, fully connected layers are often used for object recognition[14, 13], region proposal[27], etc., and transposed convolutions or dilated convolutions together with convolutional layers can be used for semantic segmentation[20, 37].

An example, LeNet-5 (1998) [14], is shown in Figure 6. 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 subregion 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.

The bottom layers in the convolutional layer stack have smaller receptive fields while 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. This trend of varying 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[12], 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

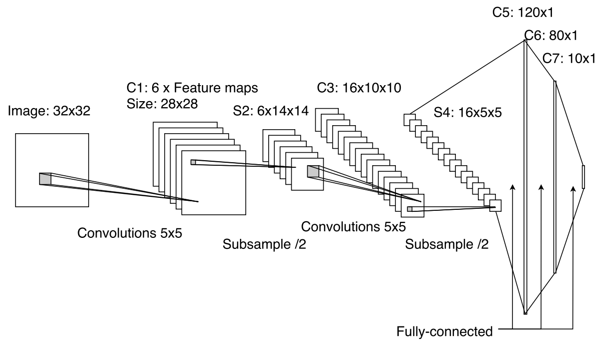

Figure 6: An example convolutional neural network, LeNet-5[14]

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 the traditional multilayer perceptron is that it is easier to optimize due to the local connectivy pattern of convolutional layers and spatial weights sharing. Because of the design choice of convolutional neurons and maximum pooling, translation invariance as well as scaling invariance and distortion invariance to some extent are achievable for convolutional neural networks.[14] Different from the traditional handcrafted features, learnable convolutional features normally generalize well and can achieve better performance for dataset with a complex input distribution.[13] 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[29], Microsoft COCO[19], 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. The success of CNN models on object classification tasks can be extended for semantic image segmentation tasks.[20] As we discussed in the

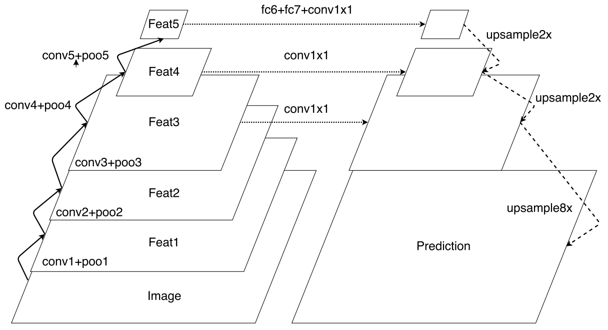

Figure 7: Fully convolutional network (FCN) by Long et al. (2015) [20].

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 for the CIFAR dataset classification.

previous session, convolutional layers can extract hierarchical features, which from low-level to high-level encode information from local to global. In contrast to object classification tasks, which normally only need global information to resolve semantics, segmentation tasks also require local information to resolve locations. One of the primary difficulties of applying CNN model to segmentation tasks is how to combine global information and local information to solve semantics and locations altogether. Long et al.[20] proposed a so-called skip architecture to aggregate information from the local low-level features in the hierarchy with global information from the high-level features. 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.

Figure 7 shows the architecture of so-called fully convolutional networks (FCN) by Long et al. (2015).

A.3 Transferring convolutional neural nets

B Deep Learning with Label Noise

B.1 Learning in the presence of label noise

NNAR, NAR, NCAR

Such a noise model is called noisy not at random (NNAR) [6] because the noise depends on not only the true label y but also the inputs x.

B.2 Deep learning models robust to label noise

C Supportive information

C.1 An 8-layer Convolutional neural network

C.2 Evaluation metrics

(overall) accuracy

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

precision

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

recall

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

f1-score

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

intersection over union (IU)

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

D Additional Results

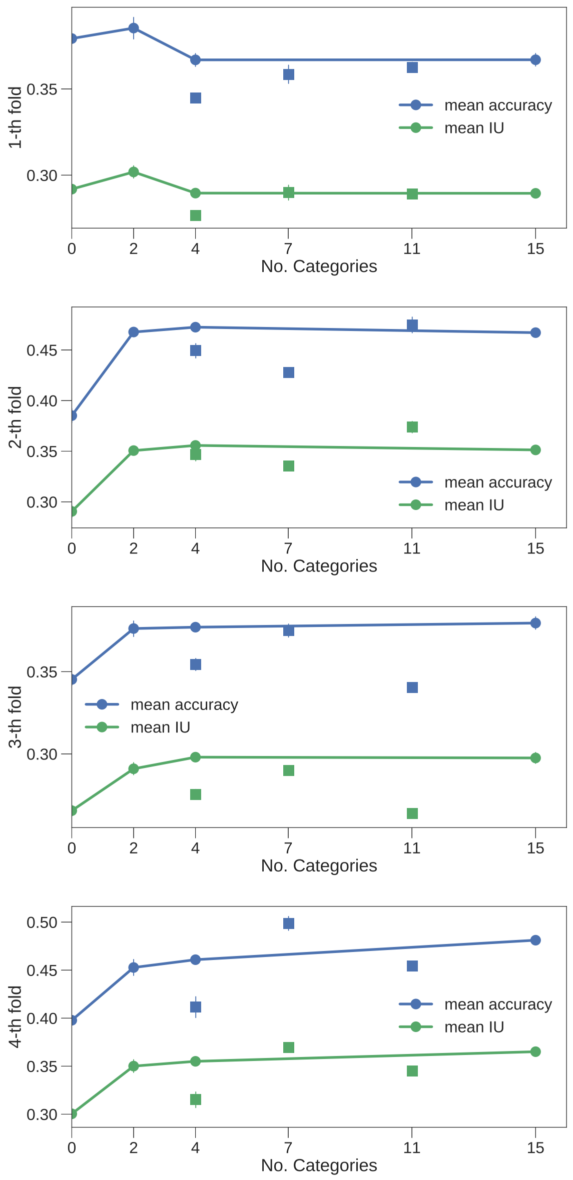

Figure 8: The influence of categorizing classes on test performance of fine-tuned models for each fold, addition to Figure 3.