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 19, 2017

Learn transferable features with noisy segmentation datasets

First Author
Institution1
Institution1 address
firstauthor@i1.org

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

August 19, 2017

Abstract

TODO: brief introduction We experimented in this work the influence of segmentation noises on transferability of pre-trained convolutional neural network (CNN) models.

1 Introduction

The state-or-the-art convolution neural nets benefits from transferring weights from convolutional neural network (CNN) models trained with a subset of images from ImageNet, referred to as the *ImageNet models*.[25, 2, 12] These ImageNet CNN models[17, 35, 37, 13] trained for object recognition task benefit from the availability of a large-scale supervised dataset, the ILSRVC dataset[33] which contains around 1.2 million labeled images. In contrast to object recognition tasks, it is difficult to collect a dataset for semantic segmentaiton on that large scale. This difficulty is natural because it costs much more efforts for people to segment than to classify an image. Therefore, scales of semantic segmentation datasets are normally much smaller than object recognition dataset. For instance, one of the largest segmentation datasets, Microsoft COCO2014[23], contains 123,287 images for 80 object categories, smaller than the ILSRVC dataset by a factor of 10 approximately. Therefore, semantic segmentation models are often trained with constraints of limited numbers of available training images. A commonly used method for improving segmentation performance in the limitation of lacking training samples is to transfer weights from the pre-trained ImageNet models.[25, 2]

However, there can be limitations for these ImageNet models to significantly improve performance for a segmentation model. Firstly, the ImageNet models were trained with relatively low resolution natural images. In some domains of interest, training images can be non-natural, for example, aerial images, images from bird's eye view, and medical images; In other domains, mages may have different lighting conditions from the ImageNet images such as photos taken in a dark warehouses; Images to be segmented may also have higher resolution than the ImageNet ones. To train a segmentation model in these domains, it can be beneficial to fine-tune the ImageNet model using a similar dataset in the domain of interest if there exists one. Secondly, the ImageNet models cannot be applied directly to RGB-D images or 3D images like CT scans and MRI scans in 3D. Lastly, segmentation models may have different design thinkings from classification models due to the inherent differences of the two tasks. For example, features' translation invariance and reduced resolution for object recognition CNN models can reduce localization accuracy for segmentation.[42, 2] The challenges in adapting ImageNet models for segmentation tasks can result in different architectures for segmentation models[42]. Therefore, a model pre-trained with segmentation datasets in a similar domain can be useful to achieve good segmentation performance with a small training set.

The pre-training segmentation datasets may, however, contain label noises, and the existence of segmentaion noises should not magnificently affect the transferability of pre-trained weights. The use of crowd-sourcing platform like Mechanical Turk is common nowadays to collect datasets on a large scale. However, 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 [23, 6], to ensure the correctness of segmentations. A slight decrease in percentage of segmentation errors, such as from 1% to 0%, may require extraordinary extra efforts due to the difficulty of identifying errors. If not requiring “gold standard” segmentations, the efforts saved for correctness can be made for segmenting more images so that the result segmentations can be larger in numbers though traded with the existence of label noises. In some domains, for example medical imaging, the “gold standard” itself can be ambiguous and cause disagreements among experts. [1] Besides, freely available labels may exist for particular tasks, as alternatives to manual annotations. But these labels often contain structural noises depending on the way they were created. For example, one can use digital maps, like OpenStreetMap, to segment aerial images. These segmentations constructed from maps would suffer from the incomplete annotation as well as registration problems.[27] Ideally, the use of these noisy datasets for pre-training should not affect the result weights transferring to another dataset. If negative influences of label noises on weights transferability were remarkable, methods of compensating the noises then become relevant.

Segmentation noises Noises of different kinds can exist in segmentation labels, for example, inexaustive segmentation, misclassification of segments, false segmentating, over-segmenting, under-segmenting, etc. In particular, we consider only noises happen to labels for the whole segments instead of individual pixels, assuming the outline of segments are always correct. That leaves us three types of noises: inexaustive segmentation, misclassification and false segmentation. Inexaustive segmention, i.e., there exists objects left unsegmented, is one of the most frequent noises. A typical situation of inexaustive segmentation is when images contain huge amounts of objects of the same kind, e.g. a flock of sheeps or a pile of products. Misclassfication of segments can be avoided to some extent by asking annotators to segment one category at a time.[23] Occasional misclassified objects may nevertheless still exist because of the ambiguity of category definations, for example, the misclassified bears and teddy bears in the Microsoft COCO dataset. False segmentation denotes objects not of interest wrongly segmented as objects of interest. It may occur due to unclear category defination, the lack of knowledge or simply visual bias. We synthesized these three types of noises with a well-annotated dataset and studied their influences to the trained weights transferability separately.

Supposing misclassification had negative influence on feature transferability, as we will show in Section 5.1, a straightforward way of correcting noisy labels is to binarize classes: positive for foreground pixels and negative for background pixels. Jain et al.[16] trained a fully convolutional network to generate foreground segmentationions, and the trained model was demonstrated generalize well for object categories not included in training. Binarizing classes can be considered as converting precise but inaccurate labels to accurate but inprecise labels. For convolutional feature pre-training, precise supervision are likely not to transfer, because transferability of features is correlated to if features are general, i.e. if they depend on particular categories.

If we consider inexhaustive segmenation only the problem becomes similar to a so-called *positive and unlabelled learning* (PU learning) setup[21]. 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 P set in an inexhaustively 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 that there is no easy way to generate reliable negative labels out of it so that the traditional semi-supervised learning techniques are 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 in the presence of inexhaustive segmentation can be thus considered as a learning problem with only positive examples and unlabeled examples.

To summary, the main topics discussed in this thesis are:

1. We investigated the influence of labels noises on feature transferability for semantic segmentation.

2. Binarizing classes for pre-training can alleviate the

[1] M: But that’s OK or not? This is what probabilities solve...

effect caused by the existence of misclassification of segments.

3. We proposed a class-dependent loss for unlabeled examples to compensate the influence of unsegmented objects of interest.

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 pre-training with noisy segmentations and how to synthesize the three types segmentation noises. We proposed the Exponential Unlabeled (EU) loss for noisy negative samples in Section 4. Experiments in Section 5.1 were designed to study the influences of inexhaustive segmentations, misclassification and false segmentation separately. The proposed exponential unlabeled loss was evaluated in synthesized PU learning setups in Section 5.2. Discussions are included in Section 6 and conclusions are summarized in Section 7.

2 Related work

Transfer Learning Weights of convolutional neural networks were proved "transferable" not only to another dataset[34, 40] but also to other applications[8, 25]. Transferring weights of CNN models for tasks and applications that are different from which the weights were originally trained with allows to improve the performance of learning without engaging in much efforts spent for data-labeling.[28] Yosinski et al.[40] reported that initializing a network with transferred features from almost any number of layers can produce a boost to the fine-tuning performance with better generalization. Transferability is possible between completely disimilar tasks, e.g. natural classes and man-made classes. It could be also possible to transfer weights trained with noisy labels.

Apart from supervised pre-training, one can also perform unsupervised learning to obtain pre-trained features typically with auto-encoders[39, 26], deep belief networks[14, 19]. Though a few studies[5, 4, 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[9] and its variants. We used random weights initialization as the lower bounds for pre-training with noisy labels. Features learned in the presence of label noises should at least outperform random weights because noisy information shoudl be still better than no information.

Deep Learning with Noisy Labels The impact of random flipped labels on classification performance has been investigated by [36, 29] 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. By contrast, Rolnick et al.[32] argued that deep neural networks can learn robustly from the noisy dataset 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.

Methods, including a linear niose layer on top of the model output[36], loss correctness with an estimation of the noise transition matrix[29] were proposed to compensate the negative effect on classification performance introduced by flipped labels. Additionally, Reed & Lee[30] proposed a bootstrapping loss to emphasize *perceptual consistency* in training when incomplete and noisy labels exist. Artificial datasets synthesized from well-annotated datasets were used to evaluate their methods. We designed our experiments using noisy segmentations synthesized with perfect segmentation as well.

Positive and Unlabeled Learning Traditional positive and unlabeled learning methods proposed for text classification[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, either naïve Bayesian (NB) or supported vector machine (SVM), with RN set and P set, while updating the RN set with a selected classifier. It is unrealistic to train iteratively a couple of deep learning models because it would take significantly longer time to train neural networks than to train a naïve Bayesian (NB) model or supported vector machine (SVM).

Alternatively, one can treat all unlabeled examples as negative and simply reweigh positive and negative examples[20]. Under the assumption of randomly labeled positive examples, Elkan & Noto[3] 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 whereas we showed that it is possible to train deep neural networks for multiple classes with only positive and unlabeled examples.

The problem of weighting unlabeled examples is that it relies on the assumption that positive examples are unlabeled at random. However, very positive examples are much more likely to be labeled in practice, because they are easier to recognize. It is therefore reasonable to instead assume that the probability of being labeled for a positive example is directly related to the posterior probability.[38] In other words, very positive prediction indicates that an unlabeled example is actually positive. Lin et al.[22] proposed a so-called Focal Loss to down-weight confident examples and thus focus training on hard negatives for imbalanced learning. Our exponential unlabeled loss follows a similar design thinking as [22] to compensate noisy negative.

3 Feature transferability with label noises

3.1 Problem Formulation

Semantic Segmentation A deep learning model for semantic segmentation normally consists of two main functions: a CNN feature extractor G that extracts hierarchical feature maps h from images x, followed by a classifier H that generates pixel-by-pixel prediction to fit labels y. Together they form a segmentation model F to predict class probabilities for each of the pixels in a given image x:

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

Feature Transferability The initialization of feature extractor G can be transferred from a pre-traine model by fine-tuning. Ideally, fine-tuning the feature extractor $\tilde{G}$ pre-trained with noisy labels $\tilde{y}$ could result in a segmentation model with equivalent performance as fine-tuning feature extractor G^* pre-trained with true labels y^*. Compared to randomly initialized feature extractor G, the fine-tuning performance improvement on test set indicates the transferability of a pre-trained feature extractor. The difference of performance improvement for fine-tuned models initialized with $\tilde{G}$ and with G^* can tell the influence of label noises on feature transferability.

3.2 Label noise synthesization

It is difficult to find a dataset with both clean and noisy labels available, so we tried to synthesize segmentation noises with well-annotated labels. A straightforward way to synthesize noisy labels is to corrupt true labels stochastically for each segment with a corruption model. The corruption model describes the probability of an observed label conditioning on the the true label, the image and binary variable for label errorness.

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 $i, j, k \in \mathrm{Z}^+$.

False segmentations In the presence of false segmentations, 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 $\sum_0^K p_k = 1$. TODO This is an emulation of xxx

Misclassification 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* [7]. This assumption does not hold in every cases of practice, for example, some instances can be more likely to be misclassified 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 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:

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

4 Positive and Unlabeled Learning

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.

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.*

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 [29]:

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

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

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 [38] for examples with negative labels:

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

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.

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 [38] to get rid of the effect of outliers. In our case, the negative examples given confident predictions by classifier can be considered as outliers.

Minimum entropy regularization and bootstrapping objective An alternative way of encouraging confident positive predictions is to introduce minimum entropy regularization[10]. Reed et al.[30] 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:

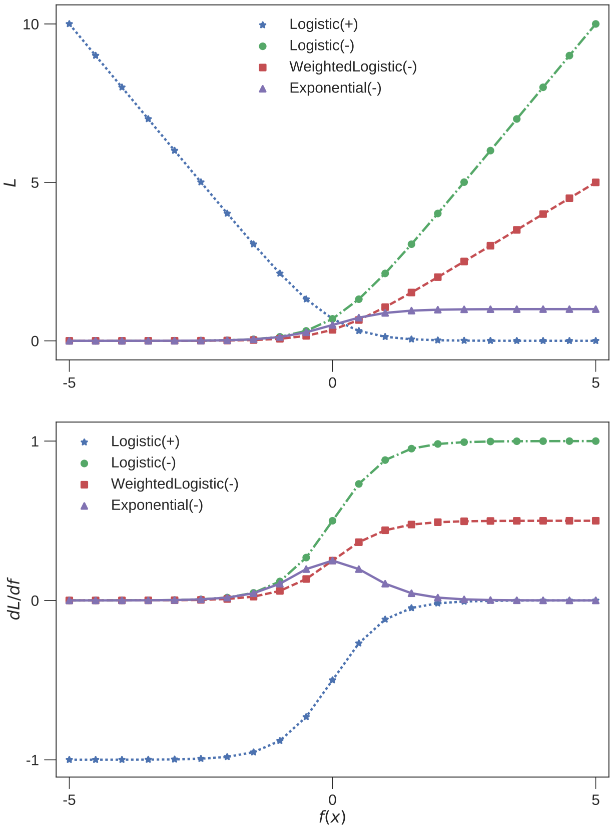


Figure 1: The Logistic Loss, Weighted Logistic Loss, Exponential Loss and their derivatives with respect to logits.

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

which intuitively encourages the model to make confident predictions[10].

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

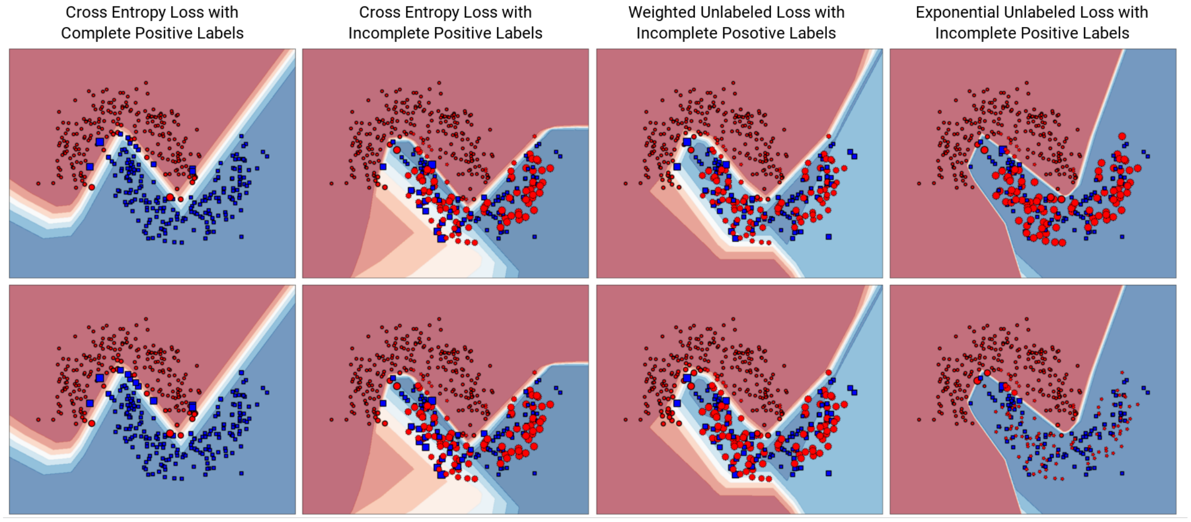


Figure 2: Decision boundaries for a 2-layer multilayer perceptron with different losses on a 2D moons dataset with unlabeled positive. Four hundreds samples per class were drawn randomly from two interleaving half circles with noises added with a minor standard deviation. A **red circle** indicates an 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.

been discussing modifications to the logistic loss for binary classification and segmentation. Similiar modifications can be applied to softmax loss as well. In a multi-class 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 3, 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 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.

5 Experiments

5.1 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[6]. 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[25] 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.[11] 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 [25] 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 3.1.

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 3.1. 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 3.1. 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.

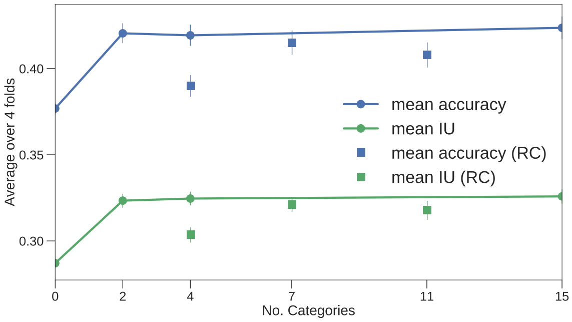


Figure 3: Test performance for fine-tuned models initialized with weights pre-trained with categorized 15 classes. Isolated error bars located aside from the lines denote random categorizations (RC) of the 15 classes. The displayed mean IU/mean accuracies and standard deviations were averaged over four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

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-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 [6] 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.

5.2 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.

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.

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 C. Each model was trained from scratch with Adam optimizer and base learning rate 0.0001. Model performances were evaluated on 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. The weighted unlabeled loss and hard bootstrap loss were able to achieve more balanced precision and recall than the exponenetial unlabeled loss. That is because the two former losses were weighted by classes frequency of observed labels, around 0.67 for the negative class and 2 for positive classes, whereas the exponential unlabeled loss was not. The reason for this is that reweighing exponential unlabeled loss by observed label frequencies would trade too much precision for recall (0.74 and 0.83 respectively), resulting in a worse f1-score than not reweighing losses. Compared to the weighted unlabeled loss, the exponential unlabeled loss seems to be more sensible to the choice of class weights and easier over-balancing due to the zero contribution to model updates for confident positive predictions. One solution to this would be adding a hyperparameter to tune the boundary between confident and unconfident predictions so that a tradeoff between precision and recall can be made in addition to changing the class weights. Recently, Lin et al. [22] proposed a Focal Loss[22] for object detection with imbalanced dataset, which takes the form of

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.	0.76	0.xx	0.xx	0.xx
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.

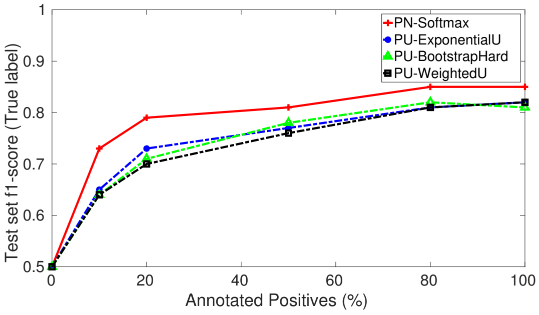


Figure 4: Varying percentage of labeled positive images. **P+N** represents training with percetage of images with reliable positive and negative labels and **P+U** stands for training with the positive (P) and unlabeled (U) sets.

$l_{y=t} = -\alpha_t(1-p(y=t|x)))^\gamma \log p(y=t|x)$. This focal loss can be considered as an attempt going towards this direction.

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.

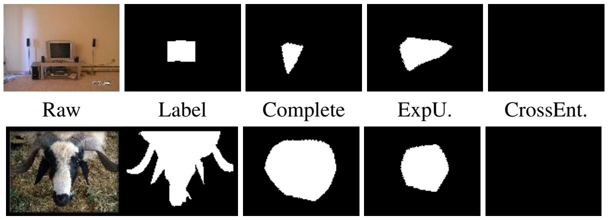
In the segmentation setup, we used again the PASCAL VOC2011 dataset with extra segmentation[11]. We synthesized inexhaustive segmentations the same way as described in Section 5.1. The same AlexNet-FCN model were trained together with the different loss functions for class 0 to predict binary segmentation, 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 5.1. 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.

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.

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 differences 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.


Raw Label Complete ExpU. CrossEnt.

Raw Label Complete ExpU. CrossEnt.

Figure 5: Selective predictions for models in Table 3.

Exponential loss help improve fine-tuning performance Additionally,

6 Discussion

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.

Binarizing/catergoring classes

- Tradeoff between precise bu inaccurate labels and accurate but inprecise labels
- The success of region proposal network is a prove of binarized labels can information about "objectness".
- Though it may depends on the scale of dataset. But since segmentation dataset is often small, it could be fine.

Upsides and downsides of exponential loss

- treat easy/hard classifications differently
- not over-punish confident positive prediction for negatively labeled examples
- easy to implement with the assumption of label noise spatial independence

Downside:

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

Future works

- Parameterizing exponential unlabeled loss to determine boundaries of confident and unconfident predictions
- Take into account label noise spatial dependence for neighboring pixels
- Experiment with over-segmentation and under-segmentation noises (negative influence expected)
- Experiment with real datasets

7 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] 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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

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

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

[37] 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.

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

[39] 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.

[40] 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.

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

[42] 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 task-specific layers.For instance, fully connected layers are often used for object recognition[18, 17], region proposal[31], etc., and transposed convolutions or dilated convolutions together with convolutional layers can be used for semantic segmentation[25, 41].

An example, LeNet-5 (1998) [18], 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[15], 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.

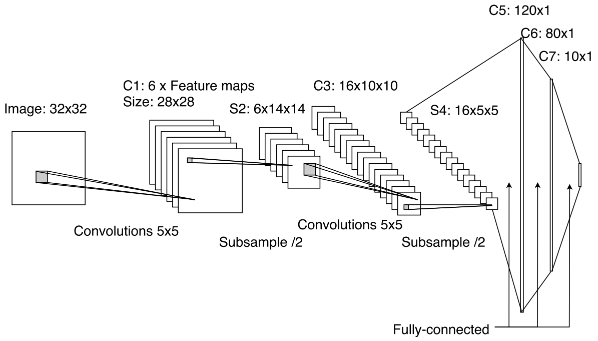


Figure 6: An example convolutional neural network, LeNet-5[18]

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.[18] Different from the traditional handcrafted features, learnable convolutional features normally generalize well and can achieve better performance for dataset with a complex input distribution.[17] 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[33], Microsoft COCO[23], 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.[25] As we discussed in the previous session, convolutional layers can extract hierarchical features, which from low-level to high-level encode

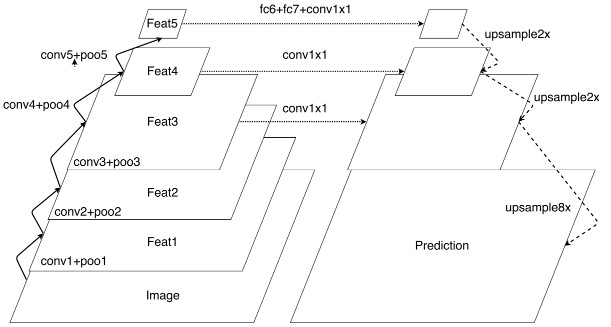


Figure 7: Fully convolutional network (FCN) by Long et al. (2015) [25].

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.[25] 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) [7] because the noise depends on not only the true label y but also the inputs x.

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.

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

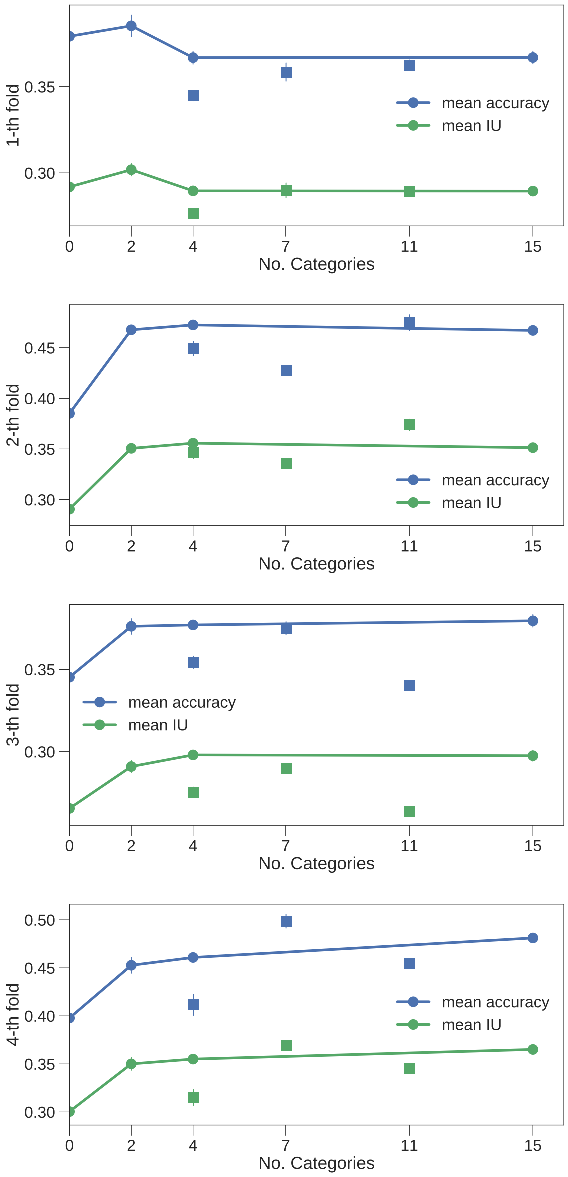


Figure 8: The influence of categorizing classes on test performance of fine-tuned models for each fold, addition to Figure 3.