Pedro H. O. Pinheiro Ronan Collobert

Idiap-RR-13-2014

AUGUST 2014

Weakly Supervised Object Segmentation with Convolutional Neural Networks

Pedro O. Pinheiro[1,2], **Ronan Collobert**[1]
[1]Idiap Research Institute, Martigny, Switzerland
[2]Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{pedro.pinheiro, ronan.collobert}@idiap.ch

Abstract

Can a machine learn how to segment different objects in real world images without having any prior knowledge about the delineation of the classes? In this paper, we demonstrate that this task is indeed possible. We address the problem by training a Convolutional Neural Networks (CNN) model with weakly labeled images, *i.e.*, images in which the only knowledge assumed on each sample is the presence or not of an object. The model, trained in an one–vs-all scheme, learns representations that distinguish image patches that belong to the class of interest from those that belong to the background. The per-pixel segmentation is obtained by applying the model to the patch surrounding the pixel and assigning the inferred class to that pixel. Our system is trained using a subset of the Imagenet dataset. The experiments are validated on two challenging classes for segmentation: cats and dogs. We show both quantitatively and qualitatively that the model achieves good accuracy results for these classes on the Pascal VOC 2012 competition, without using any prior segmentation knowledge. This model is powerful in the sense that it learns how to segment objects without the use of costly fully-labeled segmentation datasets.

1 Introduction

Object segmentation, *i.e.*, the process of selecting a set of pixels of an image such that all the pixels in the set belongs to a given object, is one of the fundamental problems in computer vision. Its main difficulty consists on the fact that each object in the world generates an infinite number of images as the position, pose, lightning, texture, geometrical form and background varies. An useful natural image segmenter might take all those variations into account and be able to segment a given object independent of them.

Much of the progress in the field was achieved by efficient feature descriptors for images such as SIFT/HOG features, bag-of-word image representations [1, 2] or deformable part models [3, 4]. Another technological advance that enabled improvement in recognition systems in natural images is the increasing computing power of machines and larger – and more realistic – datasets providing annotation for training, such as Pascal VOC [5] and Imagenet [6].

A different paradigm for object recognition systems that has been around for a while are the so called Convolutional Neural Networks (CNN) [7] models. More recently, with the increasing success of deep learning models, the interest for CNN in vision tasks has been reborn. They have been applied in different computer vision areas in the last few years, usually achieving state-of-the-art results or breaking records: 1000-category Imagenet object classification [8, 9] and detection [10], pedestrian detection [11], complete scene labeling [12, 13], multi-digit recognition [14] and face verification [15]. CNN main disadvantage, however, is the need of large number of fully-labeled

Figure 1: Cats and dogs present unconstrained variation in shape and pose, as well as a huge intra-class variation. These characteristics pose a particular challenge for segmentation.

data for training. Some tasks (*e.g.* segmentation) are much more expensive to annotate than others (*e.g.* classification), as can be seen on the different size of PascalVOC and Imagenet datasets.

In this paper, we propose a model based on CNN capable of segmenting objects that is able to (hopefully) circumvent the issue of fully-annotated data. Instead of using any segmentation (or bounding box) annotation for the training, the model is trained assuming only weakly labeled images, *i.e*, images in which the sole annotation is the class of the main object present on them. The main idea is to learn how to segment given objects by simply training a model, in an end-to-end manner, that takes as input raw data (RGB natural image) with a label of the object. This is achieved by hierarchically learning features that differentiate the class of interest and the background. Our model is trained on a large subset of the Imagenet dataset and validated on the Pascal VOC 2012 dataset (which contains segmentation annotation and thus allows quantitative analysis).

We choose two different classes to validate our model: `cat` and `dog`. These classes represent a particular challenge for classical segmentation models, as they consist of very flexible, deformable and often occluded objects. To make it even more challenging, these classes also have an enormous intra-class variation (Figure 1).

The interest of this problem is to learn where a given class of object is located in an image *with no prior knowledge of where or how the object is*. The task can be formulated as a multiple instance learning (MIL) [16]. In this formulation, no information of where the object is located is given during training. Our system learns the visual features that are present in positive images but not in negative images.

The paper is organized as follows. Section 2 presents related works. Section 3 describes the proposed strategy. Section 4 presents the results of our experiments on Pascal VOC 2012 dataset. Finally, Section 5 provides a conclusion of the paper.

2 Related Works

Parkhi *et al* [4] study a problem similar to ours: the segmentation of both `cat` and `dog` classes. The authors propose to use template-based model to detect a distinctive part for the class, followed by detecting the rest of the object via segmentation on image specific information learned from that part. Although the final objective of their work is similar to ours, their model differentiates from ours in the sense that we do not consider any annotation to achieve our results.

After the important publication of Krizhevsky *et al* [8], an increasing interest for CNN models in object recognition applications has emerged. All these models share the advantages and disadvantages of typical CNN models. Sermanet *et al* [10] present the very powerful *Overfeat* feature extractor. It consists of an improvement over [8]'s CNN model that is suitable to object classification and localization by learning to predict object boundaries. The authors achieve good results for both classification and localization. Their model is trained and validated on Imagenet and uses object localization information for a completely supervised training. Our approach differs from theirs in three important aspects: (i) we attack the much more fine-grained problem of object segmentation instead of localization, (ii) we do not use any object localization information during the training and (iii) we validate our model in a completely different dataset.

Oquab *et al* [17] shows how image representation learned from a large-scale labeled dataset (Imagenet) can be efficiently transfered to a different visual recognition task with lesser number of training data (Pascal VOC). On the same direction, Girshick *et al* [18] shows that a model trained for classification on Imagenet can be adapted for object detection in Pascal VOC. Their results are achieved by combining techniques for generating bottom-up region proposals with CNNs. The authors achieve state-of-the-art performance in object detection. They also achieve competitive results on the segmentation task with some modifications in their model.

Like us, both [17, 18] train a model on a large-scale dataset and test them on a smaller one. Unlike us, both train a network for classification on Imagenet (very similar to [8]) and modify it to adapt to the Pascal VOC taking into account the extra information of localization of object on the new dataset. These models consist of complete supervised tasks. On the other hand, we train our model for object segmentation solely on Imagenet (and considering only its weak label). No knowledge of object localization in Pascal VOC is used during the training. The second dataset is used only to validate the segmentation results.

In [19], Simonyan *et al* address the visualization of image classification models, learned using CNN. One of the two proposed visualization techniques computes a class saliency map (specific to a given image and class), which can be used for weakly supervised object segmentation, with the use of GraphCut color segmentation. This approach is similar to ours in the sense that the system learn to segment objects despite being trained on image labels only. Unfortunately this algorithm was not evaluated on public benchmarks, making a direct comparison difficult.

Finally, Song *et al* [20] also consider a problem vaguely similar to ours: localizing objects with weakly labeled data. However, they attack the simpler problem of detection instead of segmentation. Their model combines discriminative submodular cover problem for automatically discovering a set of positive object windows with a smoothed latent SVM formulation. They demonstrate relative improvement compared to previous weakly supervised object detection approach on Pascal VOC 2007.

To the best of our knowledge, this work is the first attempt to directly segment objects with only weakly supervised data.

3 Proposed Method

Convolutional Neural Network models are very powerful types of algorithms when it comes to learning representations. Its main downside, however, is that a large amount of data is necessary for these models to learn.

Among computer vision tasks, object segmentation is arguably the most expensive to label, as it requires tedious labor to precisely annotate different objects in different images at a pixel scale. On the other hand, simple object labeling is a much cheaper task. As a result, we can easily find very large scale dataset of labeled images (such as Imagenet), but not yet a large enough dataset for object segmentation.

Motivated by these previous remarks, we propose a deep learning system based on CNN that is able to segment objects and is trained only on weakly labeled data. The model is considerably simple and has a straight *end-to-end* training, unlike most of other approaches, which use engineered features, object detection results and graph cut algorithms to achieve segmentation.

As the objective of our model differs from traditional use of CNN in image tasks ([8, 10, 17]), we describe in detail our approach. We will start by describing the data used for train, then the training procedure and the approach for pixel inference in complete images during the test phase.

3.1 Data

We demonstrate the efficiency of our approach by training the model in two distinct classes, namely `cats` and `dogs`. These classes represent a particularly challenging problem in object segmentation. While they have a characteristic texture and anatomical format, they can appear in many different poses, possibly highly occluded, and also possess a very large intra-class variation.

We create a large training dataset from Imagenet containing images of both classes and also a third class labeled as `background` – set of images in which neither of the classes appear. This is important for the model to distinguish in a pixel level the difference between the foreground and

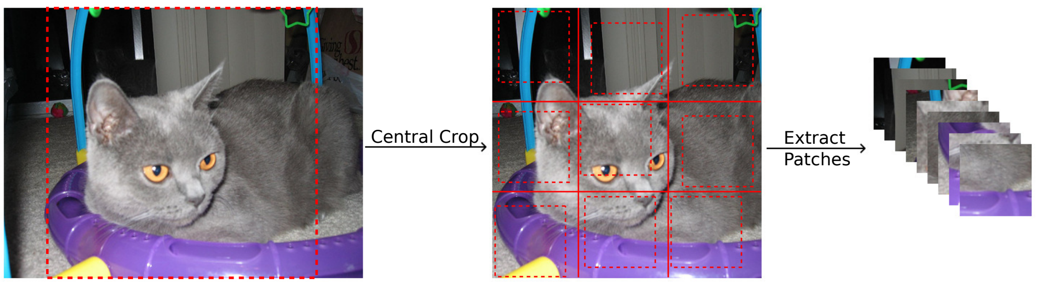

Figure 2: Each training image is scaled and cropped such that its final size is 300×300. The image is divided into N (in the example, $N = 9$) parts and a patch is randomly extracted from each part to be fed to the network.

the background. We consider all the sub-classes located below the class `cat` (`wnid` n02121620, 30 sub-classes and a total of 37345 images) and `dog` (`wnid` n02084071, 164 sub-classes and a total of 187775 images) in Imagenet. For the background, we choose a subset of Imagenet consisting of a total of around 1.2M images not containing the classes of interest.

As it is typical for convolutional network models, the size of the input image must be fixed. The variable size images from Imagenet are therefore down-sampled to a fixed 300×300 resolution. In case of rectangular images, the image is rescaled such that the shorter dimension has size 300 and then cropped out the central 300×300 patch. To increase even more the dataset, jitter is added to the image every time it is used during the training. These jitters consist of horizontal flip, rotation, scaling, brightness and contrast modification. Each image is then normalized so that their pixel values are in the range $[-1, 1]$ for each RGB channel. No other preprocessing is done during training.

3.2 Training

We train one architecture for each class in the one-vs-all scheme. Each model is trained to segment one specific class. We give the weak label $y = 1$, if the object appears in the image and 2 otherwise. Although each class is trained separately, the model for both architecture is identical.

The main idea of our approach consists on training a network to differentiate pixels as being either from foreground (`cat` or `dog` class) or background. As no segmentation (or bounding box) information is used, the model is trained in a way that it learns how to spot the differences between the foreground and the background class in RGB patches.

To achieve this objective, at each training step, the model is fed with a set of N different patches from an image $\mathbf{x}_k$ and its weak label ($y_k \in \{1, 2\}$). In order to guarantee that at least one of the patches contains a part of the foreground, they are extracted from the image in a way that they do not intersect and cover the maximum area of the image. First, the 300×300 image is divided into N sub-images of size $\frac{300}{\sqrt{N}} \times \frac{300}{\sqrt{N}}$. The patches are randomly extracted (uniform distribution) from each sub-image (see Figure 2).

The architecture of our network is illustrated in Figure 3. It is composed of five learning layers: three convolutional layers (responsible for hierarchical feature learning) and two fully-connected layers for classification. The final score of the network is computed by combining the output of each patch through a *log-sum-exp* (LSE) function, lse : $\mathbb{R}^N \to \mathbb{R}$, define as

$$\text{lse}(x_1, ..., x_N) = \log(\sum_{i=1}^{N} e^{x_i}) = x_* + \log(1 + \sum_{i \neq *} e^{x_i - x_*}) \tag{1}$$

where $x_* = \max_i \ x_i$. The right hand side trick is done to avoid the numerical overflow $\text{lse}(x_1, ..., x_N) \to \infty$ in the case where one score is much bigger than the rest.

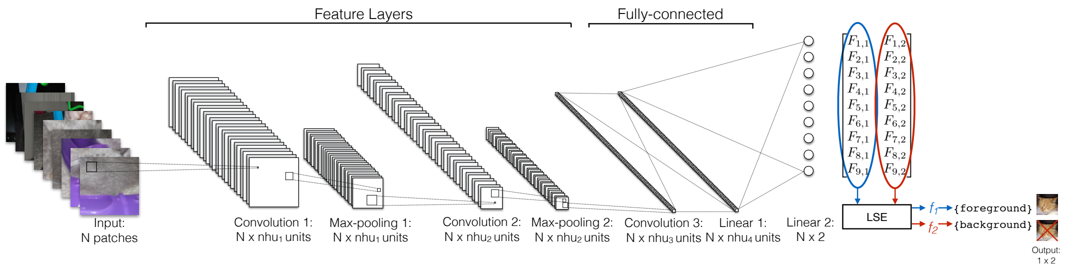
Feature Layers
Fully-connected
Input: N patches
Convolution 1: N x nhu₁ units
Max-pooling 1: N x nhu₁ units
Convolution 2: N x nhu₂ units
Max-pooling 2: N x nhu₂ units
Convolution 3: N x nhu₃ units
Linear 1: N x nhu₄ units
Linear 2: N x 2
F1,1 F2,1 F3,1 F4,1 F5,1 F6,1 F7,1 F8,1 F9,1
F1,2 F2,2 F3,2 F4,2 F5,2 F6,2 F7,2 F8,2 F9,2
LSE
f1 {foreground}
f2 {background}
Output: 1 x 2

Figure 3: Diagram of the segmentation system. N patches are randomly extracted from the image and fed to the system. After three convolutional and two fully-connected layers, the system outputs a $N \times 2$ matrix $\mathbf{F}$ which contains the score of the foreground and background for each patch. This matrix passes through a column-wise LSE function that outputs one score per class. The model is trained by maximizing the log-likelihood of the score in the LSE space through SGD.

The LSE function is increasing with respect to each argument, and convex. This function can be seen as a differentiable *soft* version of the max function. In the limit case that $x_* \gg x_i, i \neq *$, *lse* $\rightarrow$ *max*.

The intuition behind choosing LSE over max lies in the fact that, in a given training image, the class of interest might be present in more than one patch (*e.g.* in figure 2, the cat is located in various patches). LSE allows that all patches containing the object receive similar gradient during backpropagation rather than encouraging only one patch. For example, if two patches are equally likely to contain the class, they receive the same gradient.

We now describe in detail each component of the system.

The first part of the network is responsible for learning the appropriate features for distinguishing a foreground patch from a background patch. It follows the regular CNN architecture: a series of convolutional, hyperbolic tangent non-linearity and (optional) max-pooling layers. Our proposed model consists of a stack of three blocks: the first two made of a convolutional, an hyperbolic tangent and a max-pooling layer and the last just a convolutional and non-linearity layer. The network receives as input N RGB patches of fixed size, $\mathbf{X}$ (a tensor of dimension $N \times 3 \times sz \times sz$, where sz is the size of the patch which depends on the filter and pooling sizes, see Section 4). The output of the feature extraction layer $\mathbf{Y}_f(\mathbf{X})$ can be written as

$$\begin{aligned} \mathbf{Y}_f(\mathbf{X}) &= \tanh(\mathbf{W}^3\mathbf{H}^2 + \mathbf{b}^3) \\ \mathbf{H}^2 &= \tanh(\text{pool}(\mathbf{W}^2\mathbf{H}^1 + \mathbf{b}^2)) \\ \mathbf{H}^1 &= \tanh(\text{pool}(\mathbf{W}^1\mathbf{X} + \mathbf{b}^1)) \end{aligned} \tag{2}$$

where $(\mathbf{W}^i, \mathbf{b}^i)$ are the parameters of the i^{th} convolutional layer, *pool* is the max-pooling operation and *tanh* represents the hyperbolic tangent. The two fully-connected layers are then computed as $\mathbf{Y}^4_{FC} = \tanh(\mathbf{W}^4\mathbf{Y}_f(\mathbf{x}) + \mathbf{b}^4)$ and $\mathbf{F} = \tanh(\mathbf{W}^5\mathbf{Y}^4_{FC} + \mathbf{b}^5)$. The output $\mathbf{F}$ is a $N \times 2$ matrix, in which each row represents a patch and the columns represent the score for each class (foreground and background).

The final score of the network is computed by applying a LSE function at $\mathbf{F}_c$, the c^{th} column of $\mathbf{F}$

$$f_c = \text{lse}(\mathbf{F}_{1,c}, ..., \mathbf{F}_{N,c}) = \log(\sum_{i=1}^{N} e^{\mathbf{F}_{i,c}}) \tag{3}$$

where f_c represents the final score of class c ($c = 1$ for foreground or $c = 2$ for background) for the training sample $\mathbf{x}_k$.

Finally the scores are transformed into probabilities by applying a *softmax* function $p(c|f_1, f_2) = e^{f_c}/(e^{f_1} + e^{f_2})$. The model is trained by maximizing the log-likelihood of the *log-sum-exp* result for each patch using the *cross-entropy* criterion. More precisely, the parameters $(\mathbf{W}, \mathbf{b})$ of the network are learned in an end-to-end way by minimizing the negative log-likelihood over the training data

$$L(\mathbf{W}, \mathbf{b}) = -\sum_{\mathbf{x}_k} \log(p(y_k | f_1^k, f_2^k)) \tag{4}$$

where y_k represents the weak ground truth label of the training sample $\mathbf{x}_k$ ($y_k = 1$ if class of interest is present, and 2 otherwise) and, f_1^k and f_2^k represent the score of the training sample in the LSE space. The minimization is achieved with backpropagation and stochastic gradient descent (SGD) algorithm [21], considering a fixed learning rate.

After the training is finished, a second round of training was initialized to decrease the number of false positives. To this end, the segmentation model was applied to a new set of data. We infer the results (see Section 3.3) on the validation set and include all the patches in which the central pixel was falsely labeled as positive to the dataset. On the retraining, the positive sample remain the same while the negative samples were randomly chosen (with a probability of 0.5) between the original negative samples (image not containing the class) and N patches from the false positive labeled set. Note that a retraining using the misclassifications of a previous training has been shown in [22] to improve the classifier (for face detection application) .

3.3 Inference

Given a patch $\mathbf{x}_{k,(i,j)}$ of a test image $\mathbf{x}_k$ with central pixel (i, j), the network makes a first label prediction for the central pixel as

$$\hat{y}_{k,(i,j)} = \begin{cases} 1, & \text{if } \ p(c = 1 | \mathbf{x}_{k,(i,j)}, (\mathbf{W}, \mathbf{b})) > \theta \\ 2, & \text{otherwise} \end{cases} \tag{5}$$

where θ ($0 \leq \theta < 1$) is a confidence threshold for the classification. If the inferred output at a particular pixel is bigger than the threshold, it is classified as foreground object.

Extracting patches $\mathbf{x}_{k,(i,j)}$ and then feeding them through the network for all pixels of a test image is computationally very inefficient. We consider the approach proposed by [13] for an efficient inference throughout the whole test image.

Predicting the class of each pixel independently from its neighbors yields noisy predictions. We impose local regions of the same color intensities to be assigned the same label. Following the method proposed by [23], we generate superpixel segmentation of the image. To each superpixel k, we assign the class in which the majority of pixels are labeled. Figure 4 shows this procedure and illustrates the improvements of this smoothing technique.

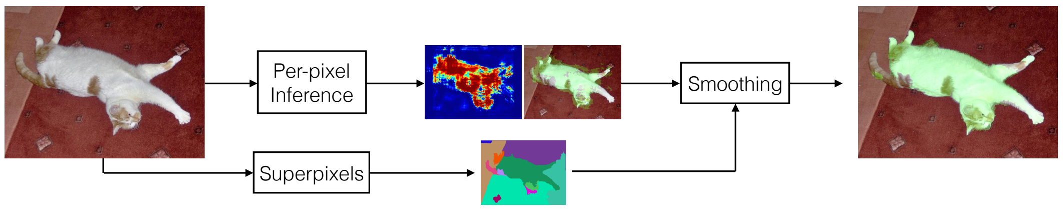

Figure 4: Example of inference in a test image. First, a per-pixel inference is made. The left image in the center shows the output score transformed into the probability of pixel (i, j) belonging to foreground (warm color means higher probability). The right one shows the per-pixel inference result. In parallel, a superpixel segmentation is computed to exploit natural contours of the image. Finally, the per-pixel inference is smoothed to give the final output. In green are the pixels labeled as ‘cat’ while the background is kept with the original color.

Table 1: Architectures used in our experiments.

	LSE₁					***LSE₂***				
Layer	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
Stage	conv+max	conv+max	conv	full	full	conv+max	conv+max	conv	full	full
# Hidden Units	40	80	160	320	2	40	80	160	320	2
Filter size	8×8	3×3	5×5	-	-	6×6	3×3	7×7	-	-
Conv. stride	1×1	1×1	1×1	-	-	1×1	1×1	1×1	-	-
Pool. size	4×4	4×4	-	-	-	8×8	2×2	-	-	-
Pool. stride	1×1	1×1	-	-	-	1×1	1×1	-	-	-
Input Size, sz	95×95	22×22	5×5	1×1	1×1	133×133	16×16	7×7	1×1	1×1
# Patches, N	9					4				
# Parameters	408922					886872				

4 Experiments

Experimental Setup We consider two different network architectures for our experiments: LSE_1 and LSE_2. Both models are as described in Section 3.2, differentiating only on the number of patches extracted per training image, N, and the size of the convolution filter and max-pooling size (and therefore the size of the input patches). Table 1 describes these architectures. Each model is validated in the two classes previously mentioned. The choice of these models represent the trade-off between two very important parameters: the number of patches that the model takes as input at each training step and the context seen by each patch. We conclude from the results that the number of patches has a more important role in this task than the context captured by each patch.

For benchmark comparison, we also show results for a simpler version of the model in which the LSE layer is replaced by a *max* operation (models max_1 and max_2). The difference between these two sets of architectures is that in the *max* layer, the forward operation consist of applying the *max* operation in each row of $\mathbf{F}$ and the backpropagation is realized only on the maximum value of each column of $\mathbf{F}$ (for each positive training sample). Results show that, as expected, the softer *Log-sum-exp* models perform better than the *max* models. This improvement is due to the fact that LSE takes into account *all* the patches containing the object of interest instead of just one.

Design architecture and hyper-parameters were chosen considering the training data of the Pascal VOC 2012 dataset. For all the experiments, the weights in the network were randomly initialized from an uniform distribution of mean zero and variance $\sigma = \sqrt{m}$, where m is the fan-in. We considered a fixed learning rate of $\lambda = 0.001$. The confidence threshold was selected through a grid search (Figure 5 shows how the AP varies in function of the confidence threshold). We pick $\theta = 0.6$ for `cat` and $\theta = 0.8$ for `dog`. All the experiments were conducted using Torch7[1].

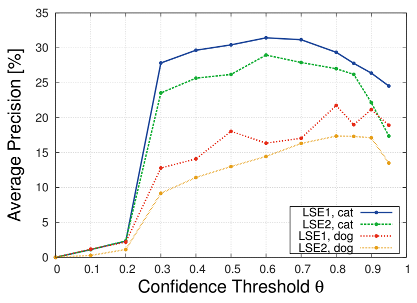

Figure 5: Average precision score vs. the confidence threshold θ for each architecture and each class.

Experimental Results Table 2 shows our results on the complete Pascal VOC 2012 validation dataset (total of 1449 images, containing 132 cats and 150 dogs). The performance of the segmentation task for each class is measured by two metrics: (i) the per-class accuracy and (ii) the mean average precision metric (AP). The former is defined by the ratio of correct classified pixels of each class (foreground and background) and the latter is defined as the number of correctly labeled pixels of that class, divided by the number of pixels labeled with that class in either the ground truth labeling or the inferred labeling[2].

We also note that our approach achieves good results for the classification task (the number of images correctly classified according to its weak label). The classification result is done by forwarding N

[1]`http://torch.ch`

[2]$\text{AP} = \frac{TruePositive}{TruePositive+FalsePositive+FalseNegative}$

Table 2: Per-class accuracy (%) for foreground/background and segmentation average precision (%) for each architecture and each class on VOC 2012 validation set.

	cat			dog		
	foreground	background	AP_{cat}	foreground	background	AP_{dog}
$\mathbf{max_1}$	32.29	99.02	22.91	56.66	96.28	19.24
$\mathbf{LSE_1}$	49.25	98.96	34.36	49.33	97.97	21.15
$\mathbf{max_2}$	40.46	97.93	21.70	40.96	96.89	16.52
$\mathbf{LSE_2}$	56.50	97.74	29.07	43.12	96.89	17.65

patches through the system and taking the maximum score as the image-level label (foreground or background). We achieve a classification score of 92.48% for `cat` and 85.70% for `dog` with the LSE_1 architecture.

The winning models of the Pascal VOC 2012 segmentation competition[3] achieve average precision accuracy of 53.5%/42.2% [24] and 49.0%/47.4% [25] for `cat`/`dog` classes, respectively. Note that these models, different from ours, were *trained with fully annotated segmentation data*. Figure 6 illustrates the output of our system in images containing the two classes despite different fur color, variety of posture, occlusion, etc.

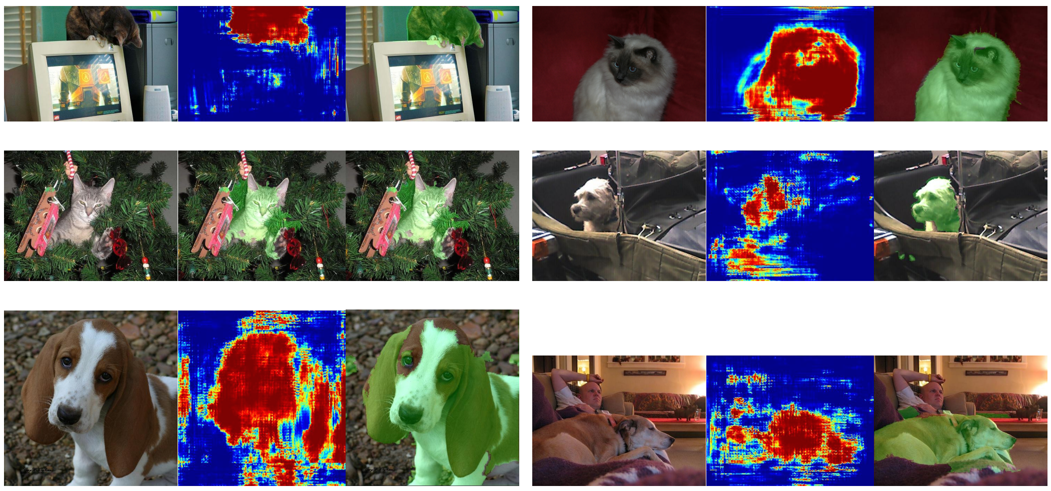
Figure 6: Cat and dog segmentation results. At each column, we show the original image, the output of the model in terms of probability of each pixel belonging to the class and the final result smoothed with superpixel.

5 Conclusion

We propose an innovative framework to segment objects with weakly supervision only. Our algorithm is able to distinguish, in a pixel level, the differences between a class of interest and the background without assuming any prior knowledge about segmentation. This is an interesting result as one might circumvent the necessity of using the very costly segmentation datasets and use only image-level annotations (which is extremely easier to acquire). We achieve promising results on a subset of classes of Pascal VOC 2012 segmentation competition consisting of two very challenging classes for the problem of segmentation. Moreover, our model is trained in a straight *end-to-end* fashion, considering as input only RGB raw pixels (contrary to competitive models, which use different features, object detection results and graph cut algorithms to achieve segmentation).

[3]`http://host.robots.ox.ac.uk:8080/leaderboard/main4.php`

References

[1] D. Larlus, J. Verbeek, and F. Jurie, “Category level object segmentation by combining bag-of-words models with dirichlet processes and random fields,” *International Journal of Computer Vision*, 2010.

[2] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2010.

[4] O. M. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman, “The truth about cats and dogs,” in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, 2010.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, 1998.

[8] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[9] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” *CoRR*, 2013.

[10] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,” in *International Conference on Learning Representations (ICLR)*, 2014.

[11] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, “Pedestrian detection with unsupervised multi-stage feature learning,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[12] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Learning hierarchical features for scene labeling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2013.

[13] P. H. O. Pinheiro and R. Collobert, “Recurrent convolutional neural networks for scene labeling,” in *International Conference on Machine Learning (ICML)*, 2014.

[14] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, “Multi-digit number recognition from street view imagery using deep convolutional neural networks,” *International Conference on Learning Representations (ICLR)*, 2014.

[15] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[16] P. M. Long and L. Tan, “Pac learning axis-aligned rectangles with respect to product distributions from multiple-instance examples.” in *Conference on Learning Theory (COLT)*, 1996.

[17] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[19] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” in *International Conference on Learning Representations (ICLR)*, 2014.

[20] H. O. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell, “On learning to localize objects with minimal supervision,” in *International Conference on Machine Learning (ICML)*, 2014.

[21] L. Bottou, “Stochastic gradient learning in neural networks,” in *Proceedings of Neuro-Nîmes*, 1991.

[22] H. A. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *IEEE Transactions On Pattern Analysis and Machine intelligence (PAMI)*, 1998.

[23] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International Journal of Computer Vision*, 2004.

[24] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, “Semantic Segmentation with Second-Order Pooling,” in *European Conference on Computer Vision (ECCV)*, 2012.

[25] W. Xia, Z. Song, J. Feng, L.-F. Cheong, and S. Yan, “Segmentation over detection by coupled global and local sparse representations,” in *European Conference on Computer Vision (ECCV)*, 2012.