Dissimilarity Coefficient based Weakly Supervised Object Detection - Supplementary Material

Aditya Arun
CVIT, KCIS
IIIT Hyderabad

C.V. Jawahar
CVIT, KCIS
IIIT Hyderabad

M. Pawan Kumar
University of Oxford,
The Alan Turing Institute

1. Learning Objective

In this section we provide detailed derivation of the objective function presented in Section 4.2 of the paper.

Given the loss function Δ (equation (6) of main paper), which is tuned for the task of object detection, we compute the diversity terms as given in equation (7) of the main paper. Recall that the diversity for any two distributions is the expected loss of the samples drawn from the two distributions. For the prediction distribution $\Pr_p$ and the conditional distribution $\Pr_c$, we derive the diversity between them and their self diversities as follows.

Diversity between prediction net and conditional net: Following equation (7) of the main paper, the diversity between prediction and conditional distribution can be written as,

$$\begin{aligned} &DIV_{\Delta}(\Pr_p, \Pr_c) \\ &= \mathbb{E}_{\mathbf{y}_p \sim \Pr_p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_p)}[\mathbb{E}_{\mathbf{y}_c \sim \Pr_c(\mathbf{y}|\mathbf{x},\mathbf{h};\boldsymbol{\theta}_c)}[\Delta(\mathbf{y}_p, \mathbf{y}_c)]]. \end{aligned} \tag{1}$$

The task specific loss function is decomposed over the bounding boxes as given in equation (5) of the main paper. We then write the expectation with respect to the conditional distribution (the inner distribution) as expectation over the random variables $\mathbf{z}$ with distribution $\Pr(\mathbf{z})$ using Law of the Unconscious Statistician (LOTUS).

$$\begin{aligned} &DIV_{\Delta}(\Pr_p, \Pr_c) \\ &= \mathbb{E}_{\mathbf{y}_p \sim \Pr_p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_p)}[\mathbb{E}_{\mathbf{z} \sim \Pr(\mathbf{z})}[\frac{1}{B}\sum_{i=1}^{B}\Delta(\mathbf{y}_p^{(i)}, \hat{\mathbf{y}}_c^{k,(i)})]]. \end{aligned} \tag{2}$$

The expectation over the random variable $\mathbf{z}$ with distribution $\Pr(\mathbf{z})$ is approximated by taking K samples from $\Pr(\mathbf{z})$,

$$\begin{aligned} &DIV_{\Delta}(\Pr_p, \Pr_c) \\ &= \mathbb{E}_{\mathbf{y}_p \sim \Pr_p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_p)}[\frac{1}{K}\sum_{k=1}^{K}\frac{1}{B}\sum_{i=1}^{B}\Delta(\mathbf{y}_p^{(i)}, \hat{\mathbf{y}}_c^{k,(i)})]. \end{aligned} \tag{3}$$

We finally compute the expectation with respect to the prediction distribution as,

$$\begin{aligned} &DIV_{\Delta}(\Pr_p, \Pr_c) \\ &= \frac{1}{BK}\sum_{i=1}^{B}\sum_{k=1}^{K}\sum_{\mathbf{y}_p^{(i)}}\Pr_p(\mathbf{y}_p^{(i)};\boldsymbol{\theta}_p)\Delta(\mathbf{y}_p^{(i)}, \hat{\mathbf{y}}_c^{k,(i)}). \end{aligned} \tag{4}$$

Self diversity for conditional net: As above, using equation (7) of the main paper, we write the self diversity coefficient of the conditional distribution as

$$\begin{aligned} &DIV_{\Delta}(\Pr_c, \Pr_c) \\ &= \mathbb{E}_{\mathbf{y}_c \sim \Pr_c(\mathbf{y}|\mathbf{x},\mathbf{h};\boldsymbol{\theta}_c)}[\mathbb{E}_{\mathbf{y}'_c \sim \Pr_c(\mathbf{y}|\mathbf{x},\mathbf{h};\boldsymbol{\theta}_c)}[\Delta(\mathbf{y}_c, \mathbf{y}'_c)]]. \end{aligned} \tag{5}$$

We now write the two expectations with respect to the conditional distribution as the expectation over the random variables $\mathbf{z}$ and $\mathbf{z}'$ respectively. The task specific loss function is decomposed over the bounding box as shown in equation (5) of the main paper. Therefore, we re-write the above equation as

$$\begin{aligned} &DIV_{\Delta}(\Pr_c, \Pr_c) \\ &= \mathbb{E}_{\mathbf{z} \sim \Pr(\mathbf{z})}[\mathbb{E}_{\mathbf{z}' \sim \Pr(\mathbf{z})}[\frac{1}{B}\sum_{i=1}^{B}\Delta(\hat{\mathbf{y}}_c^{k,(i)}, \hat{\mathbf{y}'}_c^{k,(i)})]]. \end{aligned} \tag{6}$$

In order to approximate the expectation over the random variables $\mathbf{z}$ and $\mathbf{z}'$, we use K samples from the distribution $\Pr(\mathbf{z})$ as

$$\begin{aligned} &DIV_{\Delta}(\Pr_c, \Pr_c) \\ &= \frac{1}{K}\sum_{k=1}^{K}\frac{1}{K-1}\sum_{\substack{k'=1,\\ k'\neq k}}^{K}\frac{1}{B}\sum_{i=1}^{B}\Delta(\hat{\mathbf{y}}_c^{k,(i)}, \hat{\mathbf{y}'}_c^{k',(i)}). \end{aligned} \tag{7}$$

On re-arranging the above equation, we get

$$\begin{aligned} &DIV_{\Delta}(\Pr_c, \Pr_c) \\ &= \frac{1}{K(K-1)B}\sum_{\substack{k,k'=1\\ k'\neq k}}^{K}{}^{K}\sum_{i=1}^{B}\Delta(\hat{\mathbf{y}}_c^{k,(i)}, \hat{\mathbf{y}'}_c^{k',(i)}). \end{aligned} \tag{8}$$

Self diversity for prediction net: Similar to the above two cases, using equation (7) of the main paper, we can write the self diversity of the prediction net as

$$DIV_{\Delta}(\mathrm{Pr}_p, \mathrm{Pr}_p) = \mathbb{E}_{\mathbf{y}_p \sim \mathrm{Pr}_p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_p)}[\mathbb{E}_{\mathbf{y'}_p \sim \mathrm{Pr}_p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_p)}[\Delta(\mathbf{y}_p, \mathbf{y'}_p)]]. \tag{9}$$

We then decompose the task specific loss function over the bounding boxes as described in equation (5) of the main paper,

$$DIV_{\Delta}(\mathrm{Pr}_p, \mathrm{Pr}_p) = \mathbb{E}_{\mathbf{y}_p \sim \mathrm{Pr}_p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_p)}[\mathbb{E}_{\mathbf{y'}_1 \sim \mathrm{Pr}_p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_p)}[\frac{1}{B}\sum_{i=1}^{B}\Delta(\mathbf{y}_p^{(i)}, \mathbf{y'}_p^{(i)})]] \tag{10}$$

Note that the prediction distribution is a fully factorized distribution, and we can compute its exact expectation. Therefore, we compute the two expectations with respect to the prediction distribution as,

$$\mathbb{E}_{\mathbf{y}_p \sim \mathrm{Pr}_p(\mathbf{y'}|\mathbf{x};\boldsymbol{\theta}_p)}[\frac{1}{B}\sum_{i=1}^{B}\sum_{\mathbf{y'}_p^{(i)}}\mathrm{Pr}_p(\mathbf{y'}_p^{(i)};\boldsymbol{\theta}_p)\Delta(\mathbf{y}_p^{(i)}, \mathbf{y'}_p^{(i)})] = \frac{1}{B}\sum_{i=1}^{B}\sum_{\mathbf{y}_p^{(i)}}\sum_{\mathbf{y'}_p^{(i)}}\mathrm{Pr}_p(\mathbf{y}_p^{(i)};\boldsymbol{\theta}_1)\,\mathrm{Pr}_p(\mathbf{y'}_p^{(i)};\boldsymbol{\theta}_p)\Delta(\mathbf{y}_p^{(i)}, \mathbf{y'}_p^{(i)}) \tag{11}$$

2. Optimization

2.1. Optimization over Prediction Distribution

As parameters $\boldsymbol{\theta}_c$ of the conditional distribution are constant, the learning objective of the prediction distribution (equation (13) of the main paper) results in a fully supervised training of the Fast-RCNN network [2]. Note that the only difference between training of a standard Fast-RCNN architecture and our prediction net is the use of the dissimilarity objective function (equation (13) of the main paper) instead of minimizing the multi-task loss of the Fast-RCNN.

The prediction net takes as the input an image and the K predictions sampled from the conditional net. Treating these predictions of the conditional net as the pseudo ground truth label, we compute the gradient of our dissimilarity coefficient based loss function. As the objective given in equation (13) of the main paper is differentiable with respect to parameters $\boldsymbol{\theta}_p$, we update the network by employing stochastic gradient descent.

2.2. Optimization over Conditional Distribution

A non-differentiable training procedure: The conditional net is modeled using a Discrete DISCO Net which employs a sampling step from the scoring function $S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c)$. This sampling step makes the objective function non-differentiable with respect to the parameters $\boldsymbol{\theta}_c$, even though the scoring function $S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c)$ itself is differentiable. However, as the prediction network is fixed, the above objective function reduces to the one used in Bouchacourt *et al.* [1] for fully supervised training. Therefore, similar to Bouchacourt *et al.* [1] we solve this problem by estimating the gradients of our objective function with the help of temperature parameter ϵ as,

$$\nabla_{\boldsymbol{\theta}_c} DISC^{\epsilon}_{\Delta}(\mathrm{Pr}_p(\boldsymbol{\theta}_p), \mathrm{Pr}_c(\boldsymbol{\theta}_c)) = \pm \lim_{\epsilon \to 0} \frac{1}{\epsilon}(DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_p, \mathrm{Pr}_c) - \gamma DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_c, \mathrm{Pr}_c)) \tag{12}$$

where,

$$DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_p, \mathrm{Pr}_c) = \mathbb{E}_{\mathbf{y}_p \sim \mathrm{Pr}_p(\boldsymbol{\theta}_p)}[\mathbb{E}_{\mathbf{z}_k \sim \mathrm{Pr}(\mathbf{z})}[\nabla_{\boldsymbol{\theta}_c} S_k(\hat{\mathbf{y}}_a; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) - \nabla_{\boldsymbol{\theta}_c} S_k(\hat{\mathbf{y}}_c; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c)]] \tag{13}$$

$$DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_c, \mathrm{Pr}_c) = \mathbb{E}_{\mathbf{z}_k \sim \mathrm{Pr}(\mathbf{z})}[\mathbb{E}_{\mathbf{z'}_k \sim \mathrm{Pr}(\mathbf{z})}[\nabla_{\boldsymbol{\theta}_c} S_k(\hat{\mathbf{y}}_b; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) - \nabla_{\boldsymbol{\theta}_c} S_{k'}(\hat{\mathbf{y}}'_c; \mathbf{x}, \mathbf{z'}_k, \boldsymbol{\theta}_c)]] \tag{14}$$

and,

$$\begin{aligned}
\hat{\mathbf{y}}_c &= \arg\max_{y \in \mathcal{Y}} S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \\
\hat{\mathbf{y}}'_c &= \arg\max_{y \in \mathcal{Y}} S_{k'}(\mathbf{y}; \mathbf{x}, \mathbf{z'}_k, \boldsymbol{\theta}_c) \\
\hat{\mathbf{y}}_a &= \arg\max_{y \in \mathcal{Y}} S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \pm \epsilon\Delta(\mathbf{y}_p, \hat{\mathbf{y}}_c) \\
\hat{\mathbf{y}}_b &= \arg\max_{y \in \mathcal{Y}} S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \pm \epsilon\Delta(\hat{\mathbf{y}}_c, \hat{\mathbf{y}}'_c)
\end{aligned} \tag{15}$$

In our experiments, we fix the temperature parameter ϵ as, $\epsilon = +1$.

Intuition for the gradient computation: We now present an intuitive explanation of the computation of gradient, as given in equation (12). For an input $\mathbf{x}$ and two noise samples $\mathbf{z}_k, \mathbf{z}_{k'}$, the conditional net outputs two scores $S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c)$ and $S_{k'}(\mathbf{y}; \mathbf{x}, \mathbf{z'}_k, \boldsymbol{\theta}_c)$, with the corresponding maximum scoring outputs $\hat{\mathbf{y}}_c$ and $\hat{\mathbf{y}}'_c$. The model parameters $\boldsymbol{\theta}_c$ are updated via gradient descent in the negative direction of $\nabla_{\boldsymbol{\theta}_c} DISC^{\epsilon}_{\Delta}(\mathrm{Pr}_p(\boldsymbol{\theta}_p), \mathrm{Pr}_c(\boldsymbol{\theta}_c))$.

- The term $DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_p, \mathrm{Pr}_c)$ updates the model parameters towards the maximum scoring prediction $\hat{\mathbf{y}}_c$ of the score $S_k(\mathbf{y}; \boldsymbol{\theta}_c)$ while moving away from $\hat{\mathbf{y}}_a$, where $\hat{\mathbf{y}}_a$ is the sample corresponding to the maximum loss augmented score $S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \pm \epsilon\Delta(\mathbf{y}_p, \hat{\mathbf{y}}_c)$ with respect to the fixed prediction distribution samples $\mathbf{y}_p$.

This encourages the model to move away from the prediction providing high loss with respect to the pseudo ground truth labels.

- The term $\gamma DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_c, \mathrm{Pr}_c)$ updates the model towards $\mathbf{y}_b$ and away from the $\hat{\mathbf{y}}_c$. Note the two negative signs giving the update in the positive direction. Here $\mathbf{y}_b$ is the sample corresponding to the maximum loss augmented score $S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \pm \epsilon\Delta(\hat{\mathbf{y}}_c, \hat{\mathbf{y}}')$ with respect to the other prediction $\hat{\mathbf{y}}'_c$, encouraging diversity between $\hat{\mathbf{y}}_c$ and $\hat{\mathbf{y}}'_c$.

Training algorithm for conditional net: Pseudo-code for training the conditional network for a single sample from weakly supervised data is presented in algorithm 1 below. In algorithm 1, statements 1 to 3 describe the sampling process and computing the loss augmented prediction. We first sample K different predictions $\hat{\mathbf{y}}^k_c$ corresponding to each noise vector $\mathbf{z}_k$ in statement 2. For the sampled prediction $\hat{\mathbf{y}}^k_c$ we compute the maximum loss augmented score $S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \pm \epsilon\Delta(\mathbf{y}_p, \hat{\mathbf{y}}_c)$. This is then used to find the loss augmented prediction $\hat{\mathbf{y}}_a$ given in statement 3.

In order to compute the gradients of the self diversity of conditional distribution, we need to find the maximum loss augmented prediction $\mathbf{y}_b$. Here, the loss is computed between a pair of K different predictions of the conditional net that we have already obtained. This is shown by statements 4 to 7 in algorithm 1.

For the purpose of optimizing the conditional net using gradient descent, we need to find the gradients for the objective function of the conditional net defined in equation (14) of the main paper. The computation of the unbiased approximate gradients for the individual terms in the objective function is shown in statement 8. We finally optimize the conditional net by the employing gradient descent step and updating the model parameters by descending to the approximated gradients as shown in statement 9 of algorithm 1.

3. Implementation Details

In this section, we provide additional implementation details. For the input pair $(\mathbf{x}, \mathbf{z}_k)$, the classification branch of the conditional net outputs a score function $\mathcal{G}_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c)$, which is a $B \times C$ matrix. We then sample $\hat{\mathbf{y}}^k_c$ as described in Section 3.2 of the paper. A non-maximal suppression is applied to further reduce the number of sampled bounding boxes. Corresponding to these samples, we mask the bounding box regression branch of the conditional net such that every bounding box which is not present in the sampled output $\hat{\mathbf{y}}^k_c$ is multiplied by a 0 row vector. This ensures that only those bounding boxes which are sampled by the conditional net are retained in the regression branch. The approximated gradients of the loss function is then computed and fed explicitly to the non-differentiable output branch to update the parameters of the

Algorithm 1: *Conditional net training algorithm*

Input : Training input $(\mathbf{x}, \mathbf{a}) \in \mathcal{W}$, and prediction net output $\mathbf{y}_p$

Output: $\hat{\mathbf{y}}^1_c, \ldots, \hat{\mathbf{y}}^K_c$, sample K predictions from the model

1 **for** $k = 1 \ldots K$ **do**

2 Sample noise vector $\mathbf{z}_k$, generate output $\hat{\mathbf{y}}^k_c$:

$$\hat{\mathbf{y}}^k_c = \arg\max_{y \in \mathcal{Y}} S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c)$$

3 Find loss augmented prediction $\hat{\mathbf{y}}^k_a$ w.r.t. output from prediction net $\mathbf{y}_p$:

$$\hat{\mathbf{y}}^k_a = \arg\max_{y \in \mathcal{Y}} S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \pm \epsilon\Delta(\mathbf{y}_p, \hat{\mathbf{y}}^k_c)$$

4 Compute loss augmented predictions:

5 **for** $k = 1, \ldots, K$ **do**

6 **for** $k' = 1, \ldots, K, k' \neq k$ **do**

7 Find loss augmented prediction $\hat{\mathbf{y}}^k_b$ w.r.t. other conditional net outputs $\hat{\mathbf{y}}^k_c$:

$$\hat{\mathbf{y}}^{k,k'}_b = \arg\max_{y \in \mathcal{Y}} S_k(\mathbf{y}; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \pm \epsilon\Delta(\hat{\mathbf{y}}^k_c, \hat{\mathbf{y}}')$$

8 Compute unbiased approximate gradients for $DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_c, \mathrm{Pr}_c)$ and $DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_c, \mathrm{Pr}_c)$ as:

$$\begin{aligned} &DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_p, \mathrm{Pr}_c) \\ &= \frac{1}{KB} \sum_{k=1}^{K} \sum_{i=1}^{B} \Big[\nabla_{\boldsymbol{\theta}_c} S_k(\hat{\mathbf{y}}^{(i)}_a; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \\ &\quad - \nabla_{\boldsymbol{\theta}_c} S_k(\hat{\mathbf{y}}^{(i)}_c; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \Big] \end{aligned} \tag{16}$$

$$\begin{aligned} &DIV^{\epsilon}_{\Delta}(\mathrm{Pr}_c, \mathrm{Pr}_c) \\ &= \frac{2}{K(K-1)B} \sum_{\substack{k,k'=1 \\ k' \neq k}}^{K} \sum_{i=1}^{B} \Big[\nabla_{\boldsymbol{\theta}_c} S_k(\hat{\mathbf{y}}^{(i)}_b; \mathbf{x}, \mathbf{z}_k, \boldsymbol{\theta}_c) \\ &\quad - \nabla_{\boldsymbol{\theta}_c} S_{k'}(\hat{\mathbf{y}}'^{(i)}_c; \mathbf{x}, \mathbf{z}'_k, \boldsymbol{\theta}_c) \Big] \end{aligned} \tag{17}$$

Update model parameters by descending to the approximated gradients:

$$\boldsymbol{\theta}^{t+1}_c = \boldsymbol{\theta}^t_c - \eta \nabla_{\boldsymbol{\theta}_c} DISC_{\Delta}(\mathrm{Pr}_p(\boldsymbol{\theta}_p), \mathrm{Pr}_c(\boldsymbol{\theta}_c))$$

network.

4. Experiments

4.1. Ablation Experiments

In this subsection we discuss the effects of the loss ratio and the thresholding operation on the score function for the detection task on VOC 2007 data set.

Effects of the loss ratio: The loss ratio λ, as defined in Section 4.1 of the main paper, is the ratio of the weight of the localization loss to the weight of the classification loss. In other words, with the higher the loss ratio more importance will be given by the objective function to correctly regress the bounding box labels. We choose three different loss ratios $\lambda = \{1, 0.33, 3\}$ for evaluation. The result of detection task on VOC 2007 test set are 52.1%, 51.6% and 52.4% mAP respectively. We empirically observe that assigning more weight to the localization loss helps, indicating that it is important for the networks to tweak the bounding boxes labels generated from the selective search region proposals.

Effect of thresholding the score function: As seen in Section 3.2 of the main paper, the conditional net generates samples from the score function (4) of the main paper. A low score value indicates that the conditional net is not certain of the bounding box label for an input image. Thresholding the score function would mean that we only sample bounding box labels from the conditional net when it has high certainty over the class distribution. We evaluate the result of the detection task on VOC 2007 test set for the threshold values of $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. Without any threshold, our method has a mean average precision of 51.4%. The corresponding mean average precision for the threshold values are $\{51.7\%, 52.2\%, 51.5\%, 51.0\%, 50.6\%\}$. These results indicate that it helps to apply threshold when the network is uncertain over the output classes. This is because we would not like the prediction net to learn from highly uncertain samples. We get the best results for the threshold value of 0.2. However, we also observe that choosing a large value for threshold has no effect on the detection accuracy. In this case, the network is already reasonably certain of the bounding box label, and we would not like to reject such samples.

Note that for the choice of loss ratio $\lambda = 3$ and threshold kept at 0.2, our method achieves the best detection average precision of 52.9% mAP.

4.2. Results on VOC 2012

Here, we compare our proposed method with other state-of-the art weakly supervised methods. Results for the task of detection average precision (AP) and correct localization (CorLoc) are presented in table 1 and table 2 respectively for PASCAL VOC 2012 data set. Our results are consistent with those observed for VOC 2007 data set and we get an overall increase of 1.7% over previous state-of-the-art method, W2F [8]. Our network trained and tested on a single scale outperforms W2F [8], which is trained and tested on multiple scales.

4.3. Results with Region Proposal Networks

In this subsection we show that our method extends to architectures with region proposal networks (RPN) [4], thus eliminating the need for external bounding box proposer like Selective Search [7]. This enables our framework to perform inference in real-time, while the entire pipeline is trained in an end-to-end fashion including the RPN.

For this, we replace our prediction net with Faster-RCNN [4] as shown in Figure 1. As we wish to use the same set of bounding boxes for both the networks, we share the bounding box proposals generated from RPN as shown in the figure. Furthermore, reusing the computation also makes our training efficient.

The algorithm proceeds by randomly initializing the RPN and extracting 300 bounding box proposals for each image. These proposals are then fed to the conditional net, which samples the bounding boxes corresponding to the image-level labels for the given image from the proposals. Note that as we introduce noise samples in our conditional net, we get a diverse set of sampled bounding boxes. These bounding boxes are then used to train the conditional net, which also updates the RPN thereby gradually improving the localization of the objects present in the image.

The results when using bounding box proposals from RPN is presented in Table 3. We compare the results against those achieved by using Selective Search bounding boxes. Note that, 300 bounding box proposals generated from the randomly initialized RPN has a recall rate of $44.5\% \pm 13.2\%$ on VOC 2007 data set. However, after several iterations of training, the final recall rate achieved from 300 bounding box proposals from RPN is 94.7%. This is still low when compared to the recall rate achieved by 2000 bounding box proposals from Selective Search method. We argue that due to this difference, we observe a 2% drop in accuracy. This makes a case of using more bounding box proposals for a better recall rate or using better RPN, like the one proposed in [6]. Finally, our choice of employing Faster-RCNN for the prediction net enables our framework to perform inference in real-time.

Table 1. Detection average precision (%) for different methods on VOC 2012 test set.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	pson	plant	sheep	sofa	train	tv	mAP
Jie *et al.* [3]	60.8	54.2	34.1	14.9	13.1	54.3	53.4	58.6	3.7	53.1	8.3	43.4	49.8	69.2	4.1	17.5	43.8	25.6	55	50.1	38.3
OICR [5]	71.4	69.4	55.1	29.8	28.1	55.0	57.1	24.4	17.2	59.1	21.8	26.6	57.8	71.3	1.0	23.1	52.7	37.5	33.5	56.6	42.5
W2F [8]	73.0	69.4	45.8	30.0	28.7	58.8	58.6	56.7	20.5	58.9	10.0	69.5	67.0	73.4	7.4	24.6	48.2	46.8	50.7	58.0	47.8
PredNet (VGG)	73.1	71.4	56.3	30.8	28.7	57.6	62.1	44.6	23.4	61.7	26.4	44.4	62.7	80.0	9.1	24.4	56.8	40.2	52.8	60.8	**48.4**
PredNet (Ens)	74.4	72.3	57.8	33.6	31.5	60.1	63.0	45.3	21.6	64.0	27.2	44.5	63.8	78.2	10.2	28.3	59.4	38.4	55.1	61.9	**49.5**

Table 2. CorLoc (in %) for different methods on VOC 2012 trainval set.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	pson	plant	sheep	sofa	train	tv	mean
Jie *et al.* [3]	82.4	68.1	54.5	38.9	35.9	84.7	73.1	64.8	17.1	78.3	22.5	57.0	70.8	86.6	18.7	49.7	80.7	45.3	70.1	77.3	58.8
OICR [5]	89.3	86.3	75.2	57.9	53.5	84.0	79.5	35.2	47.2	87.4	43.4	43.8	77.0	91.0	10.4	60.7	86.8	55.7	62.0	84.7	65.6
W2F [8]	88.8	85.8	64.9	56.0	54.3	88.1	79.1	67.8	46.5	86.1	26.7	77.7	87.2	89.7	28.5	56.9	85.6	63.7	71.3	83.0	69.4
Pred Net (VGG)	88.8	85.1	68.7	52.3	47.2	91.0	92.1	64.3	29.4	85.6	54.5	64.9	85.9	89.8	27.5	58.5	81.3	67.6	77.2	79.5	**69.5**
Pred Net (Ens)	89.1	87.1	70.3	54.2	49.8	92.5	92.5	64.6	25.1	87.0	54.8	60.5	88.3	85.4	32.6	62.7	83.4	63.2	79.9	81.7	**70.2**

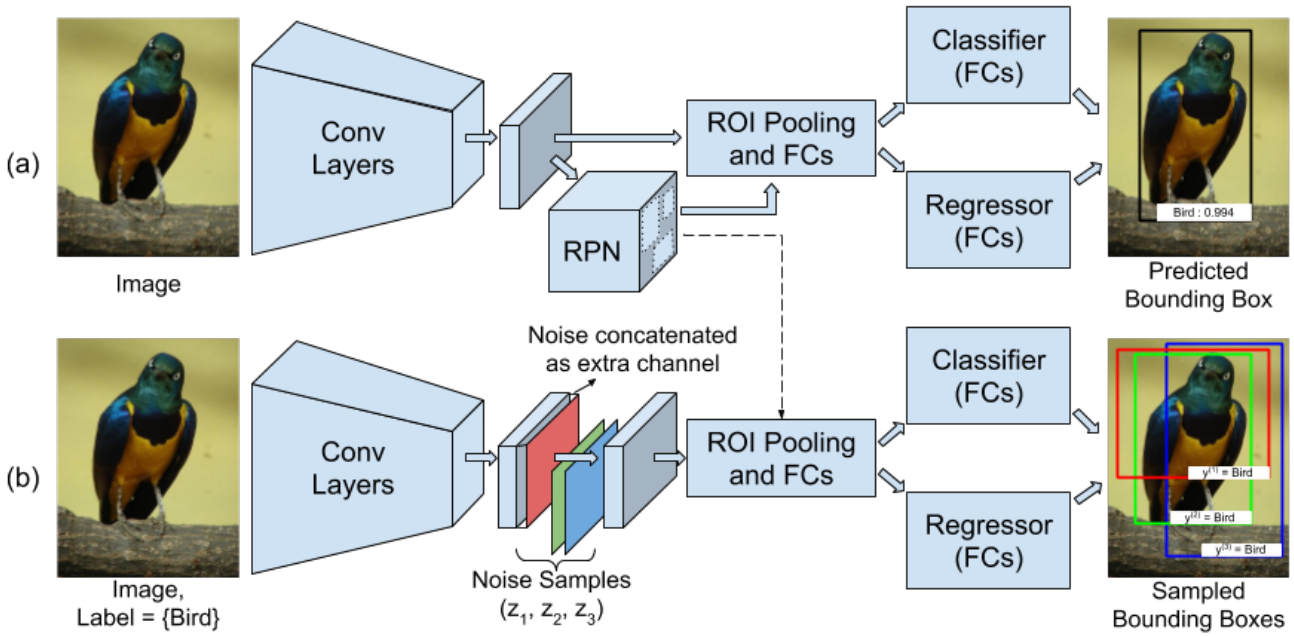

Figure 1. *The overall architecture. (a) Prediction Network: a standard Faster-RCNN architecture is used to model the prediction net. For an input image, the region proposal network (RPN) generates a set of bounding box proposals. Features from each of these proposals are computed by the region of interest (ROI) pooling layers, which are then passed through the classifier and regressor to predict the final bounding box. (b) Conditional Network: a modified Fast-RCNN architecture is used to model the conditional net. For a single input image* $\mathbf{x}$ *and three different noise samples* $\{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$ *(represented as red, green and blue matrix), three different bounding boxes* $\{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \mathbf{y}^{(3)}\}$ *are sampled for the given image-level label (bird in this example). Here the noise filter is concatenated as an extra channel to the final convolutional layer. The bounding box proposals required for the conditional net are acquired from the RPN of the prediction net. For both the networks, the initial conv-layers are fixed during training.*

	Selective Search		RPN	
	mAP %	CorLoc %	mAP %	CorLoc %
VOC 2007	52.9	70.9	50.9	69.1
VOC 2012	49.5	70.2	46.1	67.3

Table 3. Comparison of results when using bounding box proposals from Selective Search and RPN.

References

[1] Diane Bouchacourt. *Task-Oriented Learning of Structured Probability Distributions*. PhD thesis, University of Oxford, 2017.

[2] Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

[3] Zequn Jie, Yunchao Wei, Xiaojie Jin, Jiashi Feng, and Wei Liu. Deep self-taught learning for weakly supervised object localization. In *CVPR*, 2017.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[5] Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. Multiple instance detection network with online instance classifier refinement. In *CVPR*, 2017.

[6] Peng Tang, Xinggang Wang, Angtian Wang, Yongluan Yan, Wenyu Liu, Junzhou Huang, and Alan Yuille. Weakly supervised region proposal network and object detection. In *ECCV*, 2018.

[7] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013.

[8] Yongqiang Zhang, Yancheng Bai, Mingli Ding, Yongqiang Li, and Bernard Ghanem. W2F: A weakly-supervised to fully-supervised framework for object detection. In *CVPR*, 2018.