Zoom Out-and-In Network with Recursive Training for Object Proposal

Hongyang Li¹, Yu Liu², Wanli Ouyang¹ and Xiaogang Wang¹

¹ The Chinese University of Hong Kong  ² SenseTime Group Ltd.
yangli@ee.cuhk.edu.hk  liyu@sensetime.com  {wlouyang,xgwang}@ee.cuhk.edu.hk

Abstract

In this paper, we propose a zoom-out-and-in network for generating object proposals. We utilize different resolutions of feature maps in the network to detect object instances of various sizes. Specifically, we divide the anchor candidates into three clusters based on the scale size and place them on feature maps of distinct strides to detect small, medium and large objects, respectively. Deeper feature maps contain region-level semantics which can help shallow counterparts to identify small objects. Therefore we design a zoom-in sub-network to increase the resolution of high level features via a deconvolution operation. The high-level features with high resolution are then combined and merged with low-level features to detect objects. Furthermore, we devise a recursive training pipeline to consecutively regress region proposals at the training stage in order to match the iterative regression at the testing stage. We demonstrate the effectiveness of the proposed method on ILSVRC DET and MS COCO datasets, where our algorithm performs better than the state-of-the-arts in various evaluation metrics. It also increases average precision by around 2% in the detection system.

1. Introduction

Object proposal is the task of proposing a set of candidate regions or bounding boxes in an image that may potentially contain an object. In recent years, the emergence of object proposal algorithms [43, 34, 2, 44, 17, 25, 10, 4, 42] have significantly boosted the development of many vision tasks [29, 30, 6], especially for object detection [15, 7, 14, 3, 32]. It is verified by Hosang et al. [19] that region proposals with high average recall correlates well with good performance of a detector. Thus generating object proposals has quickly become the de-facto pre-processing step.

Currently, CNN models are known to be effective in generating candidate boxes [39, 28, 25]. Existing works use deep CNN features at the last layer for classifying whether a candidate box should be an object proposal. The candidate box can come from random seed [12], external boxes (selective search [43], edge box [47], etc.), or sliding windows [40]. Deep CNN-based proposal methods employ a zoom-out network, in which sub-sampling is used for reducing the resolution of features. This zoom-out design is good for image classification since sub-sampling is effective for achieving translation invariance, increasing the receptive field of features, and saving computation.

However, the zoom-out structure has two problems in object proposal for small objects. First, candidate boxes, also called anchors, are placed on the final feature map in existing works. For instance, as shown in Figure 1(b), when the down-sampling rate of the feature map is 32, moving the anchor by one step on the feature map corresponds to stepping by 32 pixels on the feature map. Due to this large stride on the image, anchors might skip small objects. Second, sub-sampling is good for translation invariance. However, it also makes it difficult to determine whether the feature is sub-sampled from foreground or background for small
objects. The lack of resolution in features is a factor that influences the ability of object proposal methods in finding small objects. As shown in Figure 1(c), if the resolution of feature map is sufficient, the anchor box with smaller stride can locate the small object. Simply increasing the resolution of image quadratically increases computation and memory, which is also limited by the current GPU memory size when a network goes deeper. Lastly, objects of different sizes are recognized with features of the same resolution, which is inappropriate.

To fit for the zoom-out network design, one could resort to features at shallow layers with higher resolution. Features from shallow layers, however, are weak in extracting high-level information that is essential for object proposal. It is verified that higher layers often carry summarized high-level semantics [46, 27]. It would be good if the high-level semantics, with sufficient resolution, can be used for finding small objects. Inspired by the hourglass network [35], we propose a zoom-out-and-in network for object proposal, namely ZIP. Figure 2 illustrates the pipeline of our algorithm. The zoom network gradually upsamples high-level feature maps to sufficient resolution. It classifies candidate region of different sizes with features of their corresponding resolutions. Such a design takes full advantage of both low-level details and high-level semantics, which is illustrated in Figure 1(d) in a nutshell. Feature 1 at shallower layer has more details about the image but cannot distinguish background from foreground. Feature 2 at deeper layer carries more semantic information, but is weak in locating small objects. If combined with Feature 3, which is deconvolved from Feature 2, the resultant Feature 4 is more useful in identifying small objects.

Iterative bounding box regression at the testing stage is found to be effective [12, 13, 11]. However, the training stage is not aware of such iterative process. Therefore, there is a mismatch in the training and testing environment for regressing the targets of bounding boxes. Gidaris et al. [12] create training samples in different regression stages by using a learned regressor in the intermediate stage or using an ideal label. However, these training samples still do not provide the learning algorithm with the actual setup at the testing stage. In this paper, we propose a new recursive training scheme where the regression targets are learned and refined consecutively during training. In this way, the iterative regression at the training stage is exactly the same as that in the testing stage. Our regressor is trained at one time and the network learns how to iteratively regress the bounding box to the ground-truth.

To sum up, our contributions in this work are as follows:

1. A zoom-out-and-in network that utilizes feature maps in different semantic levels and resolutions to detect objects of various sizes. Anchors are divided into clusters based on different scales and placed at feature maps with distinct strides. High-level semantics are gradually deconvolved and combined with low-level, high-resolution maps to help identify small objects.

2. A recursive training strategy to regress bounding box locations iteratively, where the setup of iterative regression in the training stage is consistent with that in the testing stage.

3. The proposed ZIP algorithm achieves average recall to
61.3% and 63.7% at top 500 proposals on ILSVRC DET and MS COCO, respectively. Furthermore, the proposed boxes will improve AP by around 2% for object detection compared to previous state-of-the-art.

ZIP is implemented in the Caffe [22] toolkit with Matlab wrapper and trained on multiple GPUs. The codebase and object proposal results of our method are available.

2. Related Work

The spirit of upsampling feature maps through learnable parameters is known as deconvolution, which is widely applied in different domains [33, 36, 16, 35]. Shelhamer et al. [33] first put forward a novel structure to do pixel-wise semantic segmentation via learnable deconvolution. Recently, Newell et al. [35] proposed an hourglass structure where feature maps are zoomed in and out through stacks for pose estimation. Our model is inspired by these works and yet have distinctions in several ways: First, to the best of our knowledge, our work is the first in using the deconvolution structure for object proposal. Second, existing approaches either concatenate all features [16] or use the final feature map for prediction [35], while we carefully design a network to select features at different locations (thus resolution varies) of a network for objects, e.g., low resolution features for large objects. With such a philosophy in mind, we have each object equipped with suitable features at a proper resolution.

Researchers are aware of the benefit of using features from different resolution levels. Jie et al. [23] proposed a scale-aware pixel-wise proposal framework where two separate networks are learned to handle large and small objects, respectively. Yang et al. [45] introduced a scale-dependent pooling scheme and exploited appropriate features depending on the scale of candidate proposals. Recently, Liu et al. [32] proposed a fast end-to-end learning detector where different feature layers from the VGG model output individual predictions. However, these works still use the zoom-out network structure and have limitations stated in Section 1.

Bounding box regression is widely used in object detection and region proposal [15, 14, 39, 24]. Iterative bounding box regression is recently proved to be effective at the testing stage to improve the localization accuracy in many works [13, 12, 11]. The bounding boxes approach closer to real locations step by step during iterative testing [11]. Our work in bounding box regression involves both training and testing, which is not fully investigated in previous work.

3. ZIP Algorithm

3.1. Network Architecture

Figure 2 describes the overview of the proposed zoom out-and-in network, which contains three sub-networks.

Zoom-Out Sub-network. Most existing network structures [27, 18, 41] can be viewed and used as a zoom-out network. We adopt Inception-BN [21] throughout the paper. Specifically, an image is first fed into three convolutional layers, after which the feature maps are downsampled by a rate of 8. There are nine inception modules afterwards, denoted as icp_3a-3c, icp_4a-4e and icp_5a-5b. Max-pooling is placed after icp_3c and icp_4e.

Zoom-In Sub-network. Inspired by the hourglass network [35], we adopt a zoom-in architecture to better leverage the summarized high-level feature maps for refining its low-level counterparts. The zoom-in architecture is exactly the mirrored version of the zoom-out network with max-pooling being replaced by deconvolution. We denote icp_\(x\) as the mirrored version of its counterpart icp_\(x\).

In this paper, we split the set of anchor candidates \(A\) into three clusters:

\[
A = \{A^m\}, m = 1, \ldots, 3.
\]

The scales of anchors in each level are \{16, 32\}, \{64, 128\}, \{256, 512\}, respectively. As is illustrated at the top of Figure 2, we denote the feature maps from icp_3b, icp_4d and icp_5b, as \(F^1\), \(F^2\) and \(F^3\); their mirrored counterparts from icp_3b’, icp_4d’ and icp_5b’ as \(H^1\), \(H^2\), and \(H^3\). The combination of feature maps \(F^m\) and \(H^m\) on level \(m\) are implemented via a convolution operation:

\[
G^m = \sigma(w^m_H \otimes F^m + w^m_H \otimes H^m + b^m),
\]

where \(G^m\) are the merged feature maps, \(\otimes\) denotes the convolution, \(w^m_H\) and \(w^m_H\) are the filter weights, and \(b^m\) represents the bias term.

Figure 3 visualizes feature maps at different layers in the network. \(F^1(d)\) has high resolution and provides many local details when compared with \(F^1(f)\). However, since \(F^1\) is shallow, it cannot distinguish object features from non-object features. This results in the difficulty in using \(F^1\) for the object-background classification. The upsampled feature map \(H^1\) after deconvolution is blessed with high-level semantic information for distinguishing foreground from background. After combining high-level semantic features from low-level features with local details, the resulting feature maps \(G^1(g)\) are more useful for distinguishing objects from background. Similar analysis is applied to Figure 3(e) and (h) when we identify the medium-sized helmets.

Regression and Classification Sub-network. The combined features \(G^m\) are served as input of the regression and classification sub-network. Each resolution level \(m\) has two branches, as is shown at the bottom of Figure 2. In the first branch, \(G^m\) is fed into further classification and bounding box regression, \(i.e.,\) generating object proposals from anchor set \(A^m\). After classification and regression in all resolution levels, all candidate boxes are merged and followed
loss across all levels:

\[ L = \sum_{m=1}^{M} L^m(p_t, k^*, t^*), \]  

(4)

where \( M \) is the number of resolution levels and \( N \) being the batch size.

For the RoI recursive training, let the probability estimated in the \( q \)th iteration and level \( m \) for sample \( i \) be \( \hat{p}^m_i(q) \); define the estimated region location and regression offset as \( R^m_i(q) \), \( t^m_i(q) \), respectively, then \( R^m_i(q) \) is obtained from \( R^m_i(q - 1) \) and \( t^m_i(q) \) as:

\[ R^m_i(q) = R^m_i(q - 1) + \phi(t^m_i(q)), \]  

(5)

where \( \phi(t^m_i(q)) \) is the parameter transformation function that changes the representation of \( t^m_i(q) \) defined in [14] to the one that can be used as the regression offset. Feature \( G^m_i(q - 1) \) is obtained by RoI-pooling the combined feature map \( G^m \) using region \( R^m_i(q - 1) \). The feature \( G^m_i(q - 1) \) is fed into three residual blocks for obtaining the features for \( R^m_i(q - 1) \). We use the following formulation to represent the feature extraction from residual blocks:

\[ F^m_{res,i}(q - 1) = f(G^m_i(q - 1), \theta_{res}), \]  

(6)

where \( f(\cdot) \) is the feature extraction using the residual block and \( \theta_{res} \) denotes the parameters in the residual block. The features \( F^m_{res,i}(q - 1) \) are then used for classification and bounding box regression:

\[ p_i(q) = f_c(F^m_{res,i}(q - 1), \theta_c), \]  

(7)

\[ t_i(q) = f_r(F^m_{res,i}(q - 1), \theta_r), \]  

(8)

where \( \theta_c \) and \( \theta_r \) are parameters used for classification and box regression, respectively. The process in (5)-(8) is repeated for \( Q \) times as \( q \) iterates from 1 to \( Q \) in both training and testing stages.

With the RoI recursive training included, the total loss \( L \) will be modified to:

\[ L = \sum_{m=1}^{M} \sum_{i=1}^{N} \sum_{q=1}^{Q} L^m(p_i(q), t_i(q), k^*_i(q), t^*_i(q)), \]  

(9)

where \( Q \) is the number of iterative regression during training. \( Q = 1 \) means applying regression only once. The gradient for loss in different regression stages are accumulated in a mini-batch for back-propagation.

There are several remarks regarding the training of the zoom network:

- Adjust image scale at the training stage: each training image is resized to the extent where at least one of the ground truth boxes is covered by anchors from \( A^m \). This ensures there are always positive samples in each batch.
• Control the number of negative samples in a batch. In preliminary experiments, we find training loss of the zoom network converges slowly if we fill up the rest of a batch with negative samples. This will cause the unbalance of training data for different classes when the number of positive samples is small. Thus, we strict the number of negative samples to be twice the number of positive ones.

• Additional gray category. We find adding an additional gray label ($k=2$) into training will better separate the positive from the negative. For the positive class with $k=1$, the IoU should be above 0.6; for the gray class with $k=2$, the IoU ranges from 0.35 to 0.55; and for the negative with $k=0$, the IoU is below 0.25. The number of gray samples is set to be half of the total number of positive and negative ones.

3.3. Prediction

For prediction in the zoom network, we take an inner-level and inter-scale NMS [1] scheme. Since the scale varies dynamically during training, we also forward the network in several scales, ranging from 1400 to 200 with an interval of 200. For a certain scale, we concatenate output boxes from all the levels and conduct an inner-level NMS with a threshold of 0.7; then we merge results from all scales and perform an inter-scale NMS with a threshold of 0.5.

For prediction in the RoI regression network, we average the results of bounding box regression as well as scores across all levels and scales. The box locations are replaced by the new regression results while the scores are updated by adding the regression scores. Then we conduct NMS with a threshold of 0.7 to have the final output of our ZIP algorithm.

4. Experiments

In this section, we evaluate the effectiveness of our algorithm on the problem of region proposal generation for generic objects. To this end, we evaluate our approach and compare our results with state-of-the-art object proposal methods on two challenging datasets, ILSVRC DET 2014 [8] and Microsoft COCO [31].

4.1. Datasets and setup

The Microsoft COCO 2014 dataset [31] contains 82,783 training and 40,504 validation images, where most images have various shapes surrounded by complex scenes. We use all the training images, without any data augmentation, to learn our model, and follow [37] in testing 5000 validation images for evaluation purpose (denoted as val_5k). The ILSVRC DET 2014 dataset [8] is a subset of the whole ImageNet database and consists of more than 170,000 training and 20,000 validation images. Since some training images has only one object with simple background, which has a distribution discretion with the validation set, we follow the practice of [14] and split the validation set into two parts. The training set is the train14 with 44878 images and val1 with 9205 images. We use val2 as the validation set for evaluation. Flipping each image horizontally for training is used on ILSVRC but not used on COCO.

Implementation details. We pretrain an Inception-BN [21] on the ImageNet classification dataset, which could achieve around 94% top-5 accuracy in the classification task. Inception-BN is used as the zoom-out sub-network and its mirrored version is used as the zoom-in sub-network. The three residual blocks are initialized from branch 5a to 5c of the ResNet model [18] where we change stride from 2 to 1 using hole algorithm [33]. The base learning rate is set to 0.0001 with a 50% drop every 7,000 iterations. The momentum and weight decay is set to be 0.9 and 0.0005, respectively. The maximum training iteration for both datasets is 200,000 (roughly 8 epochs) using four GPUs. The training process uses stochastic gradient descent optimization and takes around 1.5 days. We use a batch size of 300 with each class having at most 100 samples for the first branch and a batch size of 36 for the second branch. The number of iterations in the recursive regression is 2 for both training and testing if not specified. 30 anchors are used with scale increase from 16 to 512 exponentially and aspect ratio being [0.15, 0.5, 1, 2, 6.7] on ILSVRC DET and [0.11, 0.4, 1, 2, 4.3] on MS COCO.

Evaluation Metric. We use recall under different IoU thresholds and number of proposals as the main metric. The mean value of recall from IoU 0.5 to 0.95 is known as average recall (AR). AR summarizes the general proposal performance and is shown to correlate with the average precision (AP) performance of a detector better than other metrics [19]. Moreover, we compute AR of different sizes of objects to further evaluate our algorithm’s performance on a specific scale of targets. Following COCO-style evaluation, we denote three types of instance size, @small ($\alpha < 32^2$), @medium ($32^2 \leq \alpha < 96^2$) and @large ($96^2 \leq \alpha$), where $\alpha$ is the area of an object.

4.2. ILSVRC DET 2014

The top row of Figure 4 illustrates recall@prop300, @prop1500 under different IoU overlap thresholds, and recall@IoU0.5, @IoU0.8 under various number of region proposals. Our algorithm is significantly better than others for IoU threshold from 0.6 to 0.8 and the number of proposals around 300. Table 1 reports the average recall v.s. the number of proposals (from 10 to 1000) and the size of objects. Our method performs better on most metrics, es-

---

*We do not use the results directly from [12] since they tuned optimal NMS thresholds for different cases. Instead, we fix the NMS threshold as 0.5 for all the numbers of proposals through Table 1 and 2.*
especially for AR@small, where we improve by around 6% compared with [12]. This proves that our pipeline is effective in detecting small objects.

Note that the performance of our proposed method drops when the number of proposals increase at IoU@0.8. The threshold for the positive samples during training is 0.6, meaning IoU@0.8 for test is harder to align due to a mismatch with that during training; MCG [2] or selective search [43] localizes objects better when many boxes are generated, probably due to the semantic merging on a super-pixel level. Overall, the drop corresponds to a large number of proposals (typically larger than 800), and detectors will not use so many proposals in practice (300-500 are enough, where ZIP is significantly higher than others). Such a drop does not affect detection performance.

4.3. MS COCO

For MS COCO, the bottom row of Figure 4 and Table 2 shows the same evaluation metrics as those on the ILSVRC dataset. Our ZIP algorithm still outperforms most state-of-the-arts in terms of recall and AR.

4.4. Ablation study

We investigate the effectiveness of different components in our algorithm through a series of ablation studies. All the experiments use recall at IoU 0.5 (denoted as Rec@0.5) and AR of the top 300 proposals on the ILSVRC dataset.

Network design. Table 3 reports results for different network design of the zoom network. The second branch of the regression sub-network is not employed. If we only use the zoom-out sub-network and place all anchor boxes at the last feature map, the recall is 87.03%. Then we split the anchors into three groups and use features from icp_3b, icp_4d and icp_5b in the zoom-out sub-network as features of for different-sized boxes. Such a modification will increase recall by around 2%. Adding the zoom-out-and-in design further increases performance by around 2.5%. The zoom-out-and-in network increases the depth of zoom-out network and has 40 layers. By simply stacking depth of the network to 40 layers using a zoom-out design (Deeper Zoom-out + splitAnc in Table 3, we do not witness an obvious increase (90.54) compared with the zoom-out-and-in structure.

Anchor and training design. Table 4 shows recall and AR on various anchor designs and sampling schemes. Only the first branch of the regression sub-network is used. Using the original setting with 3 scales and 3 ratios as in [39], we have a 86.41% recall@0.5. By extending the number of scales to 6 and the number of aspect ratios to 5, anchor templates are increased to 30. If we linearly extend the aspect ratio to [0.25 0.5 1 2 4], recall is 89.45%, denoted by ‘30 ac.’ in Table 4. After investigating the statistics of aspect ratio in the training data, we modify the set of aspect ratio to [0.15, 0.5, 1, 2, 6.7], which is denoted as ‘30 ac. + learnStats’ in Table 4. It is observed that such a data-driven setting of aspect ratio enhances recall by 4% and 2.5% in the training and validation set, respectively. Furthermore, we find that adjusting the scale of training image (dyTrainScale), the number of negative samples in a mini-batch (ctrlNegRatio), adding an additional gray
| ILSVRC DET 2014 | AR@10 | AR@100 | AR@500 | AR@1000 | AR@Small | AR@Medium | AR@Large |
|-----------------|-------|--------|--------|---------|----------|-----------|----------|
| BING [5]        | 0.114 | 0.226  | 0.287  | 0.307   | 0.000    | 0.064     | 0.340    |
| EdgeBox [47]    | 0.188 | 0.387  | 0.512  | 0.555   | 0.021    | 0.156     | 0.559    |
| GOP [26]        | 0.208 | 0.349  | 0.486  | 0.545   | 0.022    | 0.185     | 0.482    |
| Selective Search [43] | 0.118 | 0.350  | 0.522  | 0.588   | 0.006    | 0.103     | 0.526    |
| MCG [2]         | 0.229 | 0.435  | 0.553  | 0.609   | 0.050    | 0.215     | 0.604    |
| Endres [9]      | 0.221 | 0.393  | 0.508  | 0.531   | 0.029    | 0.209     | 0.543    |
| Prims [34]      | 0.101 | 0.296  | 0.456  | 0.523   | 0.006    | 0.077     | 0.449    |
| Vigor [20]      | 0.139 | 0.325  | 0.463  | 0.551   | 0.027    | 0.092     | 0.485    |
| Faster RCNN [39] | 0.356 | 0.475  | 0.532  | 0.560   | 0.217    | 0.407     | 0.571    |
| AttractioNet [12] | 0.383 | 0.514  | 0.546  | 0.549   | 0.223    | **0.449** | 0.595    |
| ZIP w/o regress | 0.247 | 0.483  | 0.601  | 0.626   | 0.232    | 0.370     | 0.578    |
| ZIP             | **0.401** | **0.539** | **0.613** | **0.631** | **0.282** | **0.448** | **0.626** |

| MS COCO 2014    | AR@10 | AR@100 | AR@500 | AR@1000 | AR@Small | AR@Medium | AR@Large |
|-----------------|-------|--------|--------|---------|----------|-----------|----------|
| BING [5]        | 0.042 | 0.100  | 0.164  | 0.189   | 0.000    | 0.026     | 0.269    |
| EdgeBox [47]    | 0.074 | 0.178  | 0.285  | 0.338   | 0.009    | 0.086     | 0.423    |
| GOP [26]        | 0.058 | 0.187  | 0.297  | 0.339   | 0.007    | 0.141     | 0.401    |
| Selective Search [43] | 0.052 | 0.163  | 0.287  | 0.351   | 0.003    | 0.063     | 0.407    |
| MCG [2]         | 0.098 | 0.240  | 0.342  | 0.387   | 0.036    | 0.173     | 0.497    |
| Endres [9]      | 0.097 | 0.219  | 0.336  | 0.365   | 0.013    | 0.164     | 0.466    |
| DeepBox [28]    | 0.127 | 0.270  | 0.376  | 0.410   | 0.043    | 0.239     | 0.511    |
| CoGen + [17]    | 0.189 | 0.366  | 0.492  | 0.492   | 0.107    | 0.449     | 0.686    |
| DeepMask [37]   | 0.183 | 0.367  | 0.470  | 0.504   | 0.065    | 0.454     | 0.555    |
| SharpMask [38]  | 0.196 | 0.385  | 0.489  | 0.524   | 0.068    | 0.472     | 0.587    |
| AttractioNet [12] | **0.316** | **0.519** | **0.620** | **0.651** | **0.261** | **0.570** | **0.705** |
| ZIP w/o regress | 0.195 | 0.398  | 0.506  | 0.536   | 0.222    | 0.385     | 0.544    |
| ZIP             | **0.312** | **0.524** | **0.637** | **0.663** | **0.282** | **0.564** | **0.709** |

Table 1: Average recall (AR) analysis on ILSVRC val2. The AR for small, medium and large objects are computed for 100 proposals. The top two results in each metric are in **bold** and *italic*, respectively. ‘w/o regress’ means without regression.

| Structure                  | Rec@0.5 |
|----------------------------|---------|
| Zoom-out                   | 87.03   |
| Zoom-out + splitAnc        | 89.27   |
| Deeper Zoom-out + splitAnc | 90.54   |
| Zoom Out-and-In Network    | 91.76   |

Table 3: Ablation study on the zoom network structure. We use 30 anchors and treat training as a two-class problem.

| Scheme                  | Train Rec. | Rec@0.5 | AR       |
|-------------------------|------------|---------|----------|
| 9 anchors (short for ac.) | 82.02      | 86.41   | -        |
| 30 ac.                   | 84.31      | 89.45   | -        |
| 30 ac. + learnStats      | 88.24      | 91.76   | 47.54    |
| + dyTrainScale           | -          | 0.88↑   | -        |
| + ctrlNegRatio           | -          | 2.37↑   | -        |
| + grayCls                | -          | 0.31↑   | -        |
| ZIP + 30 ac. + all       | -          | **94.63**| **50.14**|

Table 4: Ablation study on the anchor design and sampling scheme. We use the zoom network in all settings. ‘××↑’ denotes absolute increase of recall by each individual strategy (see Section 4.4 for details) based on and compared to the ‘30 ac. + learnStats’ setting. ‘all’ means adopting all the strategies together.

Recursive regression design. Table 5 shows some variants of the regression sub-network structure with the second branch included. The input boxes into the RoI regression network are from the best results in Table 4, last row. First, we append only one convolution layer after the RoI-pooling layer and before the global average pooling (known as ‘simple’). Note that boxes of different sizes will use their own features at the corresponding resolution. By using a recursive regression with 2 iterations, AR boosts from 50.14% to 56.3%. If we increase depth in the recursive regression branch from one convolutional layer to three residual blocks...
Table 5: Ablation study on the design of the regression and classification sub-network structure. The repetition number for training and test are all set to be $T = 1$. See Section 4.4 for each structure’s details.

| Design          | Rec@0.5 | AR  |
|-----------------|---------|-----|
| Simple          | 93.01   | 56.30 |
| ResNet + oneStream | 95.04   | 50.54 |
| ResNet          | 92.82   | 59.45 |
| ResNet + grayCls | 93.23   | 59.48 |
| AttractioNet [12] | 92.45   | 53.93 |

Figure 5: Investigation on the number of regression repetitions. Note that $T_{\text{train}} = 1$ or $T_{\text{test}} = 1$ means applying box regression only once without recursive training.

('ResNet'), AR boosts from 56.3% to 59.45%. If we simply concatenate features from different resolutions obtained from RoI-pooling, however, AR drops by about 9%. Therefore, simple concatenation of features is worse than using features with suitable resolution. We find that adding the additional gray class during training improves performance by 0.4% (which is our final model). Note that the additional regression step could significantly increase AR from 50.14 to 59.48; but decrease recall slightly. This is because object proposals from the recursive regression will move to more confident boxes but ignore some less confident ones that may contain an object.

Number of recursive regression. Figure 5 investigates on the number of regression repetition in both training and test stage. It is observed that $T = 2$ is found to be sufficient in reaching high AR. There is a big gain between using recursive training ($T = 2$) and without ($T = 1$). More recursive iterations will slightly improve AR and yet may not be necessary considering algorithm’s efficiency. Therefore, we adopt $T = 2$ as the default setting.

To sum up, the final version of ZIP is the zoom out-and-in network with recursive regression $T = 2$, which is shown in Table 5 with gray background and denoted as ‘ResNet + grayCls’.

Table 6: Detection results on ILSVRC DET 2014: AP performance at different IoUs using R-FCN detector [7] with top 300 boxes.

| Method            | @0.50 | @0.75 | @0.5:0.95 |
|-------------------|-------|-------|-----------|
| EdgeBox [47]      | 49.94 | 35.24 | 31.47     |
| AttractioNet [12] | 50.19 | 33.45 | 32.26     |
| Selective Search [43] | 51.98 | 35.13 | 34.22     |
| ZIP               | 53.92 | 35.59 | 34.37     |

4.5. Evaluation in detection system

At last we evaluate the proposed algorithm in the detection system. We follow the same training and test procedure as that in the R-FCN detector [7]. The evaluation metric is the average precision (AP) at different IoU thresholds as well as a COCO-style AP that averages IoU thresholds from 0.5 to 0.95. Table 6 reports the detection performance of different proposal methods using top 300 boxes. It can be seen that our object proposal provides a better mAP on the ILSVRC DET dataset and are suitable for detecting objects of different sizes.

5. Conclusion

In this work, we have proposed a zoom-out-and-in network that utilizes both low-level details and high-level semantics. The information from top layers is gradually upsampled by deconvolution to reach suitable resolution for small-sized objects. The zoom-in-and-out pipeline employs features from different resolutions in a network to detect objects of various sizes. Such a strategy could alleviate the drawback of identifying small objects on feature maps with a large stride. We further propose a recursive training scheme to do multiple iterations of regression to better refine the bounding boxes, yielding a higher average recall on both ILSVRC DET and MS COCO datasets. Region proposals from the zoom network is found to provide around 2% mean AP gain for object detection when compared with other state-of-the-art object proposals.

Acknowledgement

We would like to thank S. Gidaris, X. Tong and K. Kang for helpful discussions along the way, W. Yang for proofreading the manuscript. H. Li is funded by the Hong Kong Ph.D Fellowship scheme. We are also grateful for Sense-Time Group Ltd. donating the resource of GPUs.
References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2189–2202, Nov. 2012.

[2] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[3] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016.

[4] N. Chavali, H. Agrawal, A. Mahendru, and D. Batra. Object-proposal evaluation protocol is ‘gameable’. In *CVPR*, 2016.

[5] M. Cheng, Z. Zhang, W. Lin, and P. H. S. Torr. BING: binarized normalized gradients for objectness estimation at 300fps. In *CVPR*, 2014.

[6] Z. Chi, H. Li, H. Lu, and M.-H. Yang. Dual deep network for visual tracking. *arXiv preprint*: 1612.06053, 2016.

[7] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *NIPS*, 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[9] I. Endres and D. Hoiem. Category-independent object proposals with diverse ranking. *IEEE Trans. on PAMI*, 36:222–234, 2014.

[10] A. Ghodrati, A. Diba, M. Pedersoli, T. Tuytelaars, and L. V. Gool. DeepProposals: Hunting objects and actions by cascading deep convolutional layers. *arXiv preprint*: 1606.04702, 2016.

[11] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *CVPR*, 2015.

[12] S. Gidaris and N. Komodakis. Attend Refine Repeat: Active box proposal generation via in-out localization. In *BMVC*, 2016.

[13] S. Gidaris and N. Komodakis. LocNet: Improving localization accuracy for object detection. In *CVPR*, 2016.

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

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[16] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2014.

[17] Z. Hayder, X. He, and M. Salzmann. Learning to co-generate object proposals with a deep structured network. In *CVPR*, 2016.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[19] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE Trans. on PAMI*, 2015.

[20] A. Humayun, F. Li, and J. M. Rehg. Rigid: Reusing inference in graph cuts for generating object regions. In *CVPR*, 2014.

[21] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014.

[23] Z. Jie, X. Liang, J. Feng, W. F. Hu, E. H. F. Tay, and S. Yan. Scale-aware pixelwise object proposal networks. *IEEE Trans. on Image Processing*, 25, 2016.

[24] H. Kaiming, Z. Xiangyu, R. Shaoqing, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[25] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, 2016.

[26] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *ECCV*, 2014.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.