ZoomNet: Speeding Up Single-Object Detection

Bijan Fakhri and Meredith Moore

Abstract— In this paper we attempt to trim the computational complexity associated with multi-class object detection algorithms by employing a novel neural network architecture that retroactively calculates bounding boxes of objects to solve a single-object detection problem. We introduce ZoomNet, a convolutional neural network architecture that outputs a 3D vector corresponding to the offset of the object of interest from the center of the image, and iteratively zooms into the image towards the object until the object of interest takes up the entire field of view. The corresponding field of view can be translated back into the bounding box of the object in the original image. In contrast, the current state of the art uses a semi-brute force method that queries a CNN thousands of times with subregions of the image, selecting the region with the highest confidence. Our method vastly reduces the number of times a CNN is queried. Using a synthetic dataset we produced, our approach queries the CNN an average of 45.28 times per image compared to Faster R-CNNs average 300 proposals, while obtaining an average IoU of 0.63.

Index Terms— Regional Convolutional Neural Networks, Fast- Regional Convolutional Neural Networks, Single-Object Detection,

1 Introduction

Our project began with the purpose of building a system that would detect birds using Regional Convolutional Neural Networks (R-CNNs) to eventually be used in conjunction with a laser in deterring birds from entering airfields. For this application our system would need to be capable of being realized as an embedded system. Because of this, our goal was to decrease computational resources needed and be able to do real-time detection. The majority of the current object detection (localization) literature uses a two-phased approach, where in the first phase a region proposal system proposes many (~1000 to 2000) region proposals of the image, which are then sent to the second phase, which is a classification system (usually a CNN). The classification system is used to select the region proposal with the highest confidence, giving the final bounding box of the object. Out of the 2000 region proposals sent to the classifier, only one is selected as the bounding box of the object of interest, resulting in a large amount of unnecessary computation. For single-class problems such as bird detection and pedestrian detection, this is even more wasteful.

We propose reducing the computational load by foregoing the region proposal system completely and solving the detection problem iteratively. We train a convolutional neural network to output a 3 dimensional vector <x,y,z> corresponding to the position of the object of interest to the center of the image as well as the size of that object with respect to the whole frame. This gives a rough estimate of the bounding box of the object of interest. We employ this trained CNN in an iterative fashion to hone-in on the object, increasing the accuracy of a sole iteration while vastly reducing the number of samples given to the CNN when compared to the region proposal style systems.

1.1 Related Work

Multiclass Object Detection:

Regional Convolutional Neural Networks, proposed by Ross Girshick in 2013, use a deep convolutional neural network to classify object proposals and achieve excellent object detection accuracy[1]. However, the training process of R-CNNs is complicated (uses SVMs, ConvNets, and a regressor), expensive (in both space and time), and slow (does not share computation). Fast R-CNNs were introduced in 2015 (also by Ross Girshick) and build on the R-CNN to improve training and testing speed while also increasing the accuracy[2]. Another object detection network algorithm, Faster R-CNN builds on the work of it's predecessors, however it exposes region proposal computation as a bottleneck and proposes a region proposal network that shares the convolutional features with the detection network, therefore enabling region proposal while lowering the cost[3]. While fast and faster R-CNN focus on speeding up the framework by sharing computation, they do not achieve real-time performance. Another method for real-time object detection frames object detection as a regression problem. This method You Only Look Once (YOLO) is extremely fast compared to it's R-CNN counterparts[4]. YOLO also has a smaller version of the YOLO network called Tiny YOLO, while Tiny YOLO compromises accuracy for speed, it achieves a very fast detection. Another iteration in the YOLO algorithm increases both the accuracy and the frame rate[5]. Another successful method that only looks at the image once is Single Shot Multibox Detector (SSD)[6]. SSD has achieved a higher mean average precision score than Faster R-CNN or YOLO while achieving similar speed scores to YOLO. These precision scores and frames per second scores are summarized in Table 1.

TABLE 1
COMPARISON OF MULTICLASS OBJECT DETECTION ALGORITHMS ON PASCAL VOC2007

Algorithm	mAP	FPS
Fast[2]	66.9	5
Faster[3]	73.2	7
YOLO[4]	66.4	21
Tiny YOLO	52.7	155
Fast YOLO [5]	78.6	40
SSD[6]	76.8	22

Single-Class Object Detection
For the above algorithms, the goal is real-time multiclass object detection. However, this is not exactly the goal of our proposed application. We are more interested in real-time single class detection problems like pedestrian detection, or car detection. These problems are a little bit simpler as they do not require the system to identify what the objects are, they only need to detect if the object is present, and if so, where the object is in the image. Two papers have used the R-CNN framework for pedestrian detection with varied results. In [7], they propose a scale-aware fast R-CNN (SAF R-CNN) model which has a large and small size subnetwork to better identify pedestrians of different scales. It obtains competitive, however not quite state-of-the-art results on the KITTI Pedestrian detection dataset. Another approach that uses Faster R-CNNs is found in [8]. In this paper they implemented Faster R-CNN on a pedestrian dataset and it performed sub optimally. However, somewhat surprisingly, when they used only the region proposal network, and not the down-stream classifier, they obtained better results. When coupled with boosted forests, this method obtains competitive accuracies and good speed. There are many papers that use convolutional neural networks for pedestrian detection,[9]–[13], however we do not believe that anyone has taken a similar approach to ours in a single class detection problem.

1.2 Research Question

The current state-of-the-art object detection/localization methods provide very high accuracy but suffer from huge computational costs associated with producing those accuracies. One of the constraints of those methods is that they are designed to localize and detect many classes of objects. Some applications for object localization methods do not require many-class capabilities but instead are only concerned with one class. To give an example, a system deployed at an airport tasked with detecting the presence and location of birds (to protect airplane engines) is not concerned with multi-class capabilities but most certainly must work in real-time and must be accurate. We propose a novel method by which bounding boxes of objects are calculated retroactively after iteratively zooming into an image until the object takes up the whole frame.

2 METHODS

2.1 Our Approach

In contrast to directly calculating bounding boxes such as YOLO or generating bounding proposals and sifting through them such as in region proposal networks (Fast/Faster-RCNN), our approach was to create a novel neural network architecture that outputs a 3D (x,y,z) vector corresponding to the offset of the object of interest to the center of the image. The first two components of the vector (x, y) represent the offset in the traditional dimensions of the image (height, width). The third component (z) represents the offset in the 'depth' dimension. In other words, it represents how much of the frame the object of interest takes up. This is used to infer how much we must 'zoom' into the object of interest in order to make it take up the whole frame. We call this ZoomNet. ZoomNet is designed to be used in an iterative manner, where the network is given a portion of an image (the subimage) and the network returns the 3D offset vector. The subimage is shifted and scaled according to the offset vector, giving us a new subimage. This subimage is sent into the ZoomNet like the one before it. In this way, the tradeoff between computational intensity and IoU can be directly managed by how many iterations we allow the ZoomNet to make. We are looking to further decrease the bounding box computation, while maintaining a sufficiently high bounding box accuracy, and minimizing the network memory usage. This procedure is illustrated in Figure 2 and explained below.

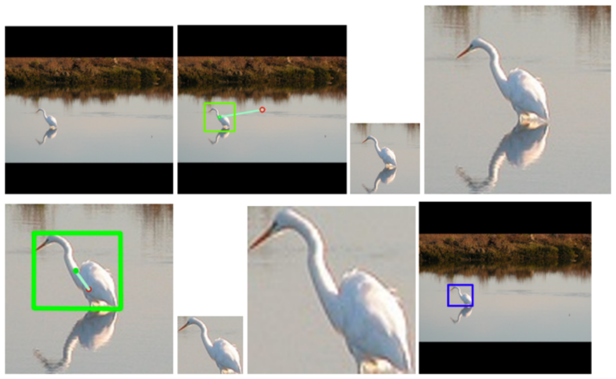

Fig 1: A demonstration of the proposed ZoomNet approach. 4 images in the top row [1, 2, 3, 4] and 4 images in the bottom row [5, 6, 7, 8].

Image 1 is the original image fed to our neural net. The neural net spits out a 3D vector, visualized in image 2 (x and y describe the position of the bird with respect to the center of the image, and z describes the size of the bounding box in relation to the whole image). After we get the 3D vector we can use it to crop the original image producing image 3. Image 3 is then blown up to the original image size (illustrated by image 4). This image is fed back into our neural network, which produces the 3D vector. Image 5 shows this vector (it is equivalent to image 2, but we are now on the second iteration). We crop image 5 producing image 6. Image 6 is blown up to the original image's (Image 1) dimensions. Feeding this image into the network gives us a 3D vector where x and y are very small (because the bird is centered) and z is close to 1 (because the bird is taking up 100% of the image). At this point we stop and we can infer the proper bounding box of the bird in the original image using the information of how we cropped and zoomed the images on each iteration.

An iterative approach is advantageous because small errors in one iteration can be smoothed out in the next iteration. This can be done by using something similar to a learning rate, where we scale how much we zoom in each iteration to be "safer". We can thus balance speed and accuracy relatively easily!

2.2 Dataset Creation

In order to simplify the problem to get a better idea of how our novel architecture was working, we developed a synthetic dataset. In this synthetic dataset, we placed orbs (concentric circles of different colors) in the images at different location and scales. This is a synthetic problem that is useful in testing the architecture's ability to localize objects without having to worry about how hard the objects are to discriminate against the background. The generated dataset featured orbs with uniformly distributed position and scale. An example of this dataset is shown in Figure 2. This orb dataset was useful in determining what architecture was best to use for our neural network.

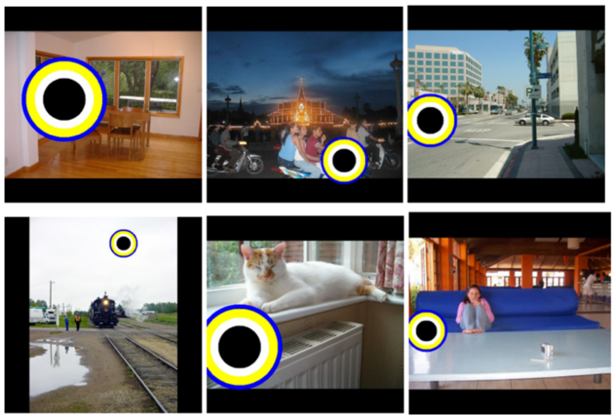

Fig 2: Examples from the toy dataset that was created to test the efficacy of the idea behind ZoomNet. In this dataet the idea is for the algorithm to find the orbs.

2.3 ZoomNet Architecture

We began the experiment by attempting to train neural network with one convolutional layer and one fully connected layer with sigmoid activation. This only learned the mean of the labels and did not work for our procedure. We then explored an architecture featuring two convolution layers (each followed by a max pooling layer) and two fully connected layers. This resulted in a considerable increase in accuracy (from basically zero to tangible numbers). Furthermore, after meeting with Dr. Baoxin Li, he recommended that we go deeper. We explored deeper and deeper architectures, optimizing for validation accuracy, and settled architecture depicted in Fig 3.

During the design process, it was important to keep in mind that the receptive field of the convolutional layers did not become excessively large in relationship to the original image. Thus, during the iterative designing phase of our architecture, we contrained the filter sizes, strides, and pooling so that the receptive field remained small.

This was to ensure that locality of features was not obscured, which would destroy the network's ability to extract information about the location of the object of interest.

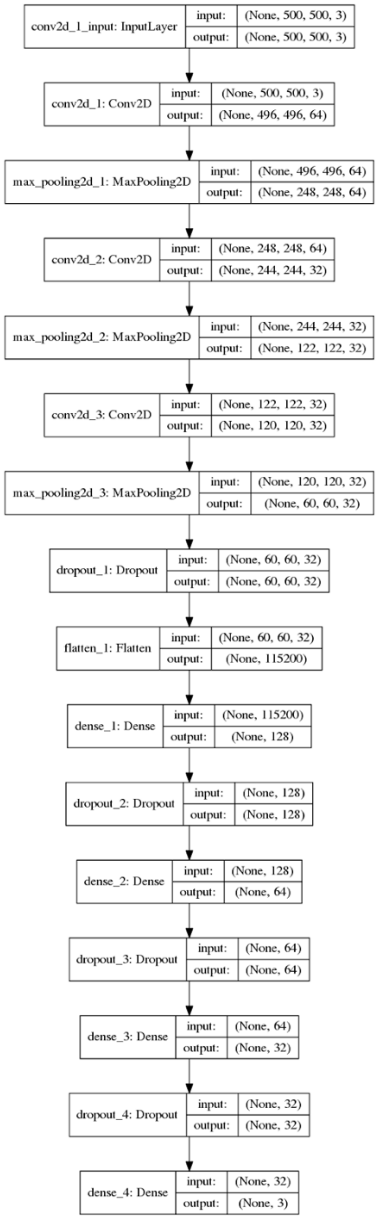
conv2d_1_input: InputLayer | input: (None, 500, 500, 3) | output: (None, 500, 500, 3)
conv2d_1: Conv2D | input: (None, 500, 500, 3) | output: (None, 496, 496, 64)
max_pooling2d_1: MaxPooling2D | input: (None, 496, 496, 64) | output: (None, 248, 248, 64)
conv2d_2: Conv2D | input: (None, 248, 248, 64) | output: (None, 244, 244, 32)
max_pooling2d_2: MaxPooling2D | input: (None, 244, 244, 32) | output: (None, 122, 122, 32)
conv2d_3: Conv2D | input: (None, 122, 122, 32) | output: (None, 120, 120, 32)
max_pooling2d_3: MaxPooling2D | input: (None, 120, 120, 32) | output: (None, 60, 60, 32)
dropout_1: Dropout | input: (None, 60, 60, 32) | output: (None, 60, 60, 32)
flatten_1: Flatten | input: (None, 60, 60, 32) | output: (None, 115200)
dense_1: Dense | input: (None, 115200) | output: (None, 128)
dropout_2: Dropout | input: (None, 128) | output: (None, 128)
dense_2: Dense | input: (None, 128) | output: (None, 64)
dropout_3: Dropout | input: (None, 64) | output: (None, 64)
dense_3: Dense | input: (None, 64) | output: (None, 32)
dropout_4: Dropout | input: (None, 32) | output: (None, 32)
dense_4: Dense | input: (None, 32) | output: (None, 3)

Fig 3: ZoomNet's Architecture

2.4 Problems Encountered

At first, we had implemented a relatively naïve network consisting of solely one convolutional layer and one fully connected layer employing sigmoid activation. This network was solely learning the mean of the labels instead of their relationship to the input image. This was solved by both adding another fully connected layer, and removing the sigmoid activation from the last fully connected layer (we realized this nonlinearity was unnecessarily squashing the outputs). After this, the network began successfully learning the relationship between the input image and the label vector.

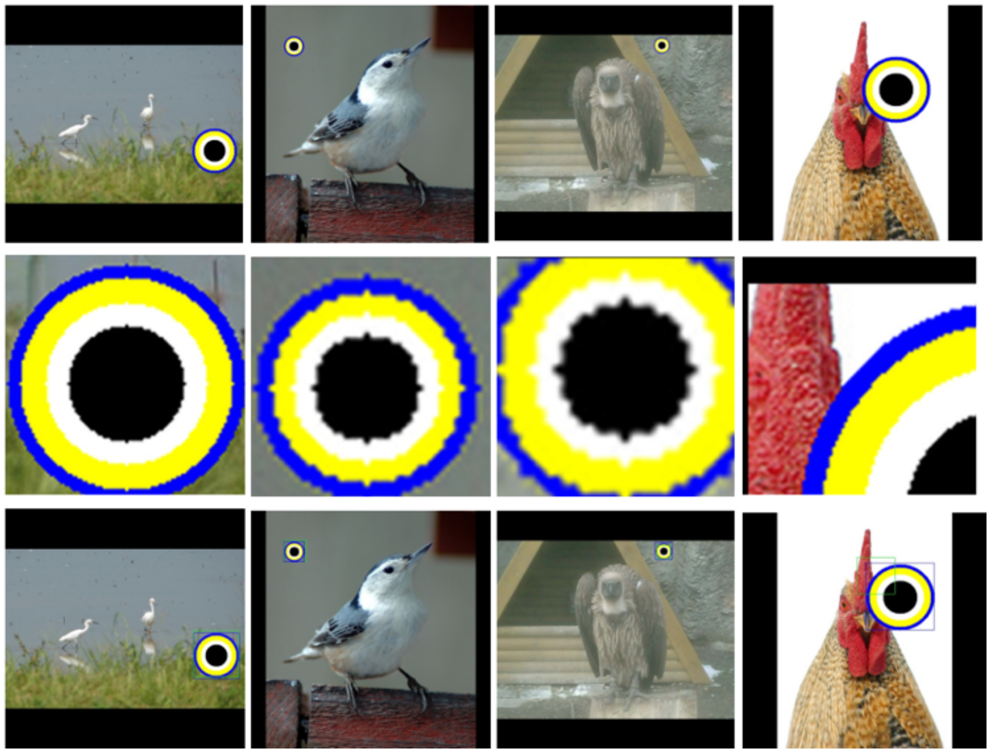
Fig 4: ZoomNet performs localization through an iterative zoom process. Each column shows an example of the zooming process with varying results. The first column shows a success case with an IoU of 0.907. The second column shows another relatively successful trial with an IoU of 0.762. The third case shows a mediocre result with an IoU of 0.70. The last column shows a failure case with an IoU of 0.174.

During the validation phase of experimentation, where the system was given an image and it iteratively found the bounding box, ~10% of the images would lock up the system. The root of the problem was discovered to be an oscillation in output values. For example, the subimage passed into the CNN would output a vector <-1, 0, z> (for simplicity of the example, we ignore the scale factor) corresponding to a shift in the negative horizontal direction (left) to center the object of interest. The system would shift the image to the left, pass it back into the CNN and receive the label <+1, 0, z>. After shifting again, the CNN would output the label <-1, 0, z>. This would result in oscillatory behavior known as live-lock. Once discovered, we remedied this by simply limiting the number of iterations the algorithm could spend consecutively shifting or zooming the image. Setting this limit solved the live-lock problem.

Training stability remained an issue throughout the project. Training accuracy and loss were difficult to interpret because they did not have a direct relationship to IoU scores, which was our target metric. For this reason, we were forced to train the same network architecture several times with several different hyper-parameters in order to train a network that would yield good results in practice (when used iteratively) . This is likely a result of the scale component of the output 'z' being significantly harder to train than the position components x and y.

3 Results

We developed a novel algorithm for obtaining bounding boxes for single-class object detection. This algorithm is discussed throughout the paper, but demonsrated in pseudocode in Figure 5. In lines 7-9 of the algorithm we initialize the return variables, and then in line 10 we send the image to ZoomNet to get a prediction. The whileloop on line 12 begins the iterative process that iterates until the object fills the whole frame. In lines 14-16, we shift the image by x and y and then get the next prediction. In lines 17-18, once the object is centered, we zoon into the object by a scalar that is inversely proportional to z.

Algorithm 1 ZoomNet
```
1:  $ZoomLimit := 20$
2:  $ShiftLimit := 20$
3:  $zNum := 50$
4:  $zDem := 50$
5:  procedure FINDOBJECT(MODEL ZOOMNET, IMAGE IMG)
6:      $ShiftLimit := 20$
7:      $xNet := 0;$
8:      $yNet := 0;$
9:      $zNet := 0;$
10:     $< x, y, z > := ZoomNet.Predict(img);$
11:     $zoomCount := 0;$
12:     while (z <FullFrame) and (zoomCount <ZoomLimit) do
13:         while ((abs(x) + abs(y) >1) and (shiftCount <ShiftLimit)) do
14:             $img := shiftImage(img, x, y);$
15:             $xNet += x * zNet;$
16:             $yNet += y * zNet;$
17:             $< x, y, z > := ZoomNet.Predict(img);$
18:             $shiftcount++;$
19:     $return < xNet, yNet, zNet >;$
```
Fig 5: ZoomNet Pseudocode

ZoomNet obtained an average IoU of 0.65. The iterative zoom process worked to locate the object of interest with varying results. Figure 4 shows the original image in the first row, the fully zoomed subimage in the second row, and the bounding box obtained from the fully zoomed subimage.

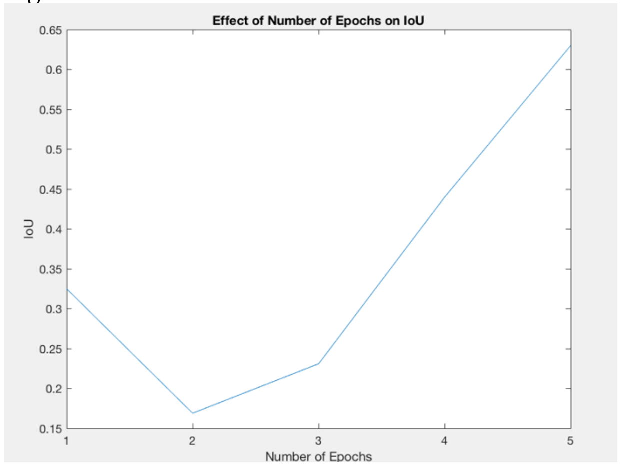

Fig 6: Effect of the number of epochs used in training on the IoU.

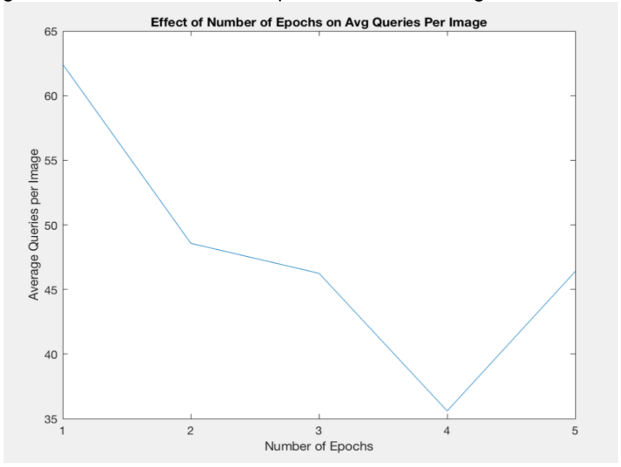

Fig 7: Effect of the number of epochs used in training on the average number of queries per image.

Figures 6 and 7 show the effects of the number of epochs that we trained the model on on both the average IoU of the model and on the average number of CNN queries per image.

4 Discussion

It is evident from Figures 6 and 7 that the tested parameters do not directly correlate to the number of Epochs the network was trained on. However, we can see a relative positive trend in IoU with increased epochs. While training the network for 6 epochs resulted in the highest average IoU, the correlation between epochs and IoU was not strong.

Figure 5 details the effect of the number of epochs which our model was trained on the number of queries to the CNN per image. This graph shows a general downward slope indicating that there is a slight correlation with the number of queries necessary as the number of epochs increases.

Our iterative localization method showed some promising results. We were able to achieve a respectable average IoU of 0.63 with an average queries per image of 46.4. While the IoU is less, the average queries per image is significantly better than some of the popular methods like Fast R-CNN.

Currently our implementation has only been tested on single-instance problems, where there is a single instance of an object of interest in the image. More than one instance could result in instability, although, we predict the case where the more salient of the instances will be picked up by our system and it will hone-in on that instance.

Another limitation of our system is that during training, the loss function is not directly correlated to the performance of the network in practice (during iterative localization). A loss function that corresponds to this performance would greatly improve training stability.

5 Future Directions

While ZoomNet is showing promising results on our synthetic dataset, we have yet to test ZoomNet on any benchmarking datasets. This would be our first next step in validating our algorithm. We would like to test on each independent class in the Pascal VOC dataset, as well as on other single-object detection datasets, namely other pedestrian detection datasets such as KITTI, ETH, Caltech Pedestrian Dataset, INRIA, or TUD-Brussels. Before we can expect ZoomNet to do well on the pedestrian datasets, we need to incorporate fixes for some of the problems that we ran into.

One of these is the many-instance problem. We propose, but have not yet implemented, a technique that would solve the many instance problem. Assuming that we are correct in predicting that instability will not arise when our system is presented with many instances of interest, it would simply be a matter of detecting the most salient object, calculating the bounding box, masking out that part of the image, and sending the image back into the system to detect the remaining objects. This would effectively solve the many-instance problem. Finding a loss function that directly correlates to the model's performance during iterative localization is a logical next step to improve this algorithm. This would likely solve the training stability problem explained in the discussion section above.

Another instance that we need to account for and test is the no-instance case. Right now our system is working under the assumption that there is an instance of the object of interest in the image. We need to test our algorithm on a set of data that has images that have the instace but also has images that do not have the instance present.

As with testing our own synthetic dataset, in testing ZoomNet on these datasets, we intend to fine tune our architecture for these benchmarking datasets.

6 Conclusion

We implemented a novel approach to single object detection that takes an iterative zoom approach to retroactively calculate bounding boxes around the object of interest.

We showed that we can reduce the number of CNN queries we have to make and still get reasonable IoU. Finding a better loss function would likely further improve the number of CNN queries we make AND improve IoU.

Future studies include implementing solutions to several problems that our algorithm faces as well as testing our solution on benchmarking single-object detection datasets.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," ArXiv13112524 Cs, Nov. 2013.
[2] R. Girshick, "Fast R-CNN," presented at the Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.
[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[5] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," ArXiv161208242 Cs, Dec. 2016.
[6] W. Liu et al., "SSD: Single Shot MultiBox Detector," ArXiv151202325 Cs, vol. 9905, pp. 21–37, 2016.
[7] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan, "Scale-aware Fast R-CNN for Pedestrian Detection," ArXiv151008160 Cs, Oct. 2015.
[8] L. Zhang, L. Lin, X. Liang, and K. He, "Is Faster R-CNN Doing Well for Pedestrian Detection?," ArXiv160707032 Cs, Jul. 2016.
[9] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson, "Real-Time Pedestrian Detection With Deep Network Cascades," 2015.
[10] W. Ouyang and X. Wang, "A discriminative deep model for pedestrian detection with occlusion handling," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3258–3265.
[11] W. Ouyang and X. Wang, "Joint Deep Learning for Pedestrian Detection," in 2013 IEEE International Conference on Computer Vision, 2013, pp. 2056–2063.
[12] Y. Tian, P. Luo, X. Wang, and X. Tang, "Pedestrian Detection aided by Deep Learning Semantic Tasks," ArXiv14120069 Cs, Nov. 2014.
[13] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," Signal Process. Image Commun., vol. 47, pp. 482–489, Sep. 2016.