Dynamic Zoom-in Network for Fast Object Detection in Large Images

Mingfei Gao[1] Ruichi Yu[1] Ang Li[2*] Vlad I. Morariu[3*] Larry S. Davis[1]
[1]University of Maryland, College Park [2]DeepMind [3]Adobe Research
{mgao,richyu,lsd}@umiacs.umd.edu anglili@google.com morariu@adobe.com

Abstract

We introduce a generic framework that reduces the computational cost of object detection while retaining accuracy for scenarios where objects with varied sizes appear in high resolution images. Detection progresses in a coarse-to-fine manner, first on a down-sampled version of the image and then on a sequence of higher resolution regions identified as likely to improve the detection accuracy. Built upon reinforcement learning, our approach consists of a model (R-net) that uses coarse detection results to predict the potential accuracy gain for analyzing a region at a higher resolution and another model (Q-net) that sequentially selects regions to zoom in. Experiments on the Caltech Pedestrians dataset show that our approach reduces the number of processed pixels by over 50% without a drop in detection accuracy. The merits of our approach become more significant on a high resolution test set collected from YFCC100M dataset, where our approach maintains high detection performance while reducing the number of processed pixels by about 70% and the detection time by over 50%.

1. Introduction

Most recent convolutional neural network (CNN) detectors are applied to images with relatively low resolution, *e.g.*, VOC2007/2012 (about 500×400) [12, 13] and MS COCO (about 600×400) [26]. At such low resolutions, the computational cost of convolution is low. However, the resolution of everyday devices has quickly outpaced standard computer vision datasets. The camera of a 4K smartphone, for instance, has a resolution of 2,160×3,840 pixels and a DSLR camera can reach 6,000×4,000 pixels. Applying state-of-the-art CNN detectors directly to those high resolution images requires a large amount of processing time. Additionally, the convolution output maps are too large for the memory of current GPUs.

Prior works address some of these issues by simplifying the network architecture [14, 41, 9, 23, 38] to speed up detection and reduce GPU memory consumption. However, these models are tailored to particular network structures and may not generalize well to new architectures. A more general direction is treating the detector as a black box that is judiciously applied to optimize accuracy and efficiency. For example, one could partition an image into sub-images that satisfy memory constraints and apply the CNN to each sub-image. However, this solution is still computationally burdensome. One could also speed up detection process and reduce memory requirements by running existing detectors on down-sampled images. However, the smallest objects may become too small to detect in the down-sampled images. Object proposal methods are the basis for most CNN

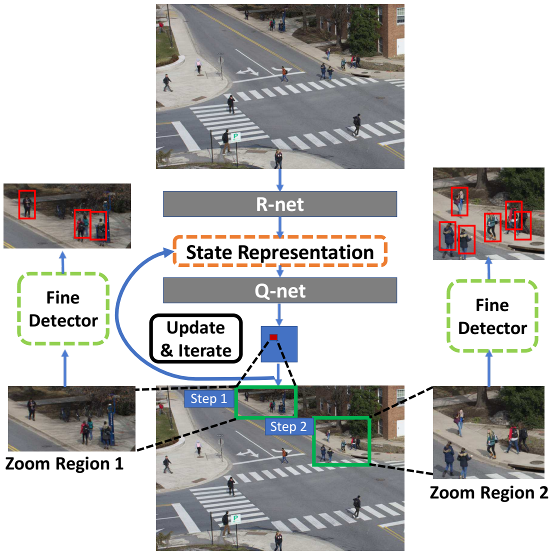

Figure 1: Illustration of our approach. The input is a down-sampled version of the image to which a coarse detector is applied. The R-net uses the initial coarse detection results to predict the utility of zooming in on a region to perform detection at higher resolution. The Q-net, then uses the computed accuracy gain map and a history of previous zooms to determine the next zoom that is most likely to improve detection with limited computational cost.

[*] The work was done while the author was at the University of Maryland

detectors, restricting expensive analysis to regions that are likely to contain objects of interest [11, 35, 44, 43]. However, the number of object proposals needed to achieve good recall for small objects in large images is prohibitively high which leads to huge computational cost.

Our approach is illustrated in Fig. 1. We speed up object detection by first performing coarse detection on a downsampled version of the image and then sequentially selecting promising regions to be analyzed at a higher resolution. We employ reinforcement learning to model long-term reward in terms of detection accuracy and computational cost and dynamically select a sequence of regions to analyze at higher resolution. Our approach consists of two networks: a zoom-in accuracy gain regression network (R-net) learns correlations between coarse and fine detections and predicts the accuracy gain for zooming in on a region; a zoom-in Q function network (Q-net) learns to sequentially select the optimal zoom locations and scales by analyzing the output of the R-net and the history of previously analyzed regions.

Experiments demonstrate that, with a negligible drop in detection accuracy, our method reduces processed pixels by over 50% and average detection time by 25% on the Caltech Pedestrian Detection dataset [10], and reduces processed pixels by about 70% and average detection time by over 50% on a high resolution dataset collected from YFCC100M [21] that has pedestrians of varied sizes. We also compare our method to recent single-shot detectors [32, 27] to show our advantage when handling large images.

2. Related work

CNN detectors. One way to analyze high resolution images efficiently is to improve the underlying detector. Girshick [16] speeded up the region proposal based CNN [17] by sharing convolutional features between proposals. Ren *et al.* proposed Faster R-CNN [33], a fully end-to-end pipeline that shares features between proposal generation and object detection, improving both accuracy and computational efficiency. Recently, single-shot detectors [27, 31, 32] have received much attention for real-time performance. These methods remove the proposal generation stage and formulate detection as a regression problem. Although these detectors performed well on PASCAL VOC [12, 13] and MS COCO [26] datasets, which generally contain large objects in images with relatively low resolution, they do not generalize as well on large images with objects of variable sizes. Also, their processing cost increases dramatically with image size due to the large number of convolution operations.

Sequential search. Another strategy to handle large image sizes is to avoid processing the entire image and instead investigate small regions sequentially. However, most existing works focus on mining informative regions to improve detection accuracy without considering computational cost. Lu *et al.* [28] improve localization by adaptively focusing on subregions likely to contain objects. Alexe *et al.* [1] sequentially investigated locations based on what has been seen to improve detection accuracy. However, the proposed approach introduces a large overhead leading to long detection time (about 5s per object class per image). Zhang *et al.* [42] improved the detection accuracy by penalizing the inaccurate location of the initial object proposals, which introduced more than 15% overhead to detection time.

A sequential search process can also make use of contextual cues from sources, such as scene segmentation. Existing approaches have explored this idea for various object localization tasks [8, 37, 30]. Such cues can also be incorporated within our framework (*e.g.*, as input to predicting the zoom in reward). However, we focus on using only coarse detections as a guide for sequential search and leave additional contextual information for future work. Other previous work [25] utilizes a coarse-to-fine strategy to speed up detection, but this work does not select promising regions sequentially.

Reinforcement learning (RL). RL a is popular mechanism for learning sequential search policies, as it allows models to consider the effect of a sequence of actions rather than individual ones. Ba *et al.* use RL to train a attention based model in [3] to sequentially select most relevant regions for object recognition and Jie *et al.* [20] select regions for localization in a top-down search fashion. However, these methods require a large number of selection steps and may lead to long running time. Caicedo *et al.* [7] designed an active detection model for object localization, which utilizes Deep Q Networks (DQN) [29] to learn a long-term reward function to transform an initial bounding box sequentially until it converges to an object. However, as reported in [7], the box transformation takes about 1.5s detection time on a typical Pascal VOC image which is much slower than recent detectors [33, 27, 32]. In addition, [7] does not explicitly consider selection cost. Although, RL implicitly forces the algorithm to take a minimum number of steps, we need to explicitly penalize cost since each step can yield a high cost. For example, if we do not penalize cost, the algorithm will tend to zoom in on the whole image. Existing works have proposed methods to apply RL in cost sensitive settings [18, 22]. We follow the approach of [18] and treat the reward function as a linear combination of accuracy and cost.

3. Dynamic zoom-in network

Our work employs a coarse-to-fine strategy, applying a coarse detector at low resolution and using the outputs of this detector to guide an in-depth search for objects at high resolution. The intuition is that, while the coarse detector will not be as accurate as the fine detector, it will identify image regions that need to be further analyzed, incurring the

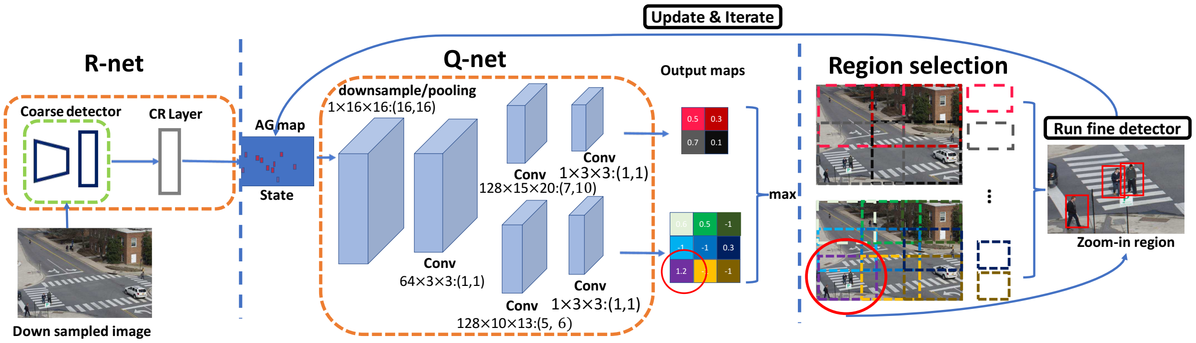

Figure 2: Given a down-sampled image as input, the R-net generates an initial accuracy gain (AG) map indicating the potential zoom-in accuracy gain of different regions (initial state). The Q-net is applied iteratively on the AG map to select regions. Once a region is selected, the AG map will be updated to reflect the history of actions. For the Q-net, two parallel pipelines are used, each of which outputs an action-reward map that corresponds to selecting zoom-in regions with a specific size. The value of the map indicates the likelihood that the action will increase accuracy at low cost. Action rewards from all maps are considered to select the optimal zoom-in region at each iteration. The notation 128×15×20:(7,10) means 128 convolution kernels with size 15×20, and stride of 7/10 in height/width. Each grid cell in the output maps is given a unique color, and a bounding box of the same color is drawn on the image to denote the corresponding zoom region size and location.

cost of high resolution detection only in promising regions. We make use of two major components: 1) a mechanism for learning the statistical relationship between the coarse and fine detectors, so that we can predict which regions need to be zoomed in given the coarse detector output; and 2) a mechanism for selecting a sequence of regions to analyze at high resolution, given the coarse detector output and the regions that have already been analyzed by the fine detector. Our pipeline is illustrated in Fig. 2. We learn a strategy that models the long-term goal of maximizing the overall detection accuracy with limited cost.

3.1. Problem formulation

Our work is formulated as a Markov Decision Process (MDP) [6]. At each step, the system observes the current state, estimates potential cost-aware rewards of taking different actions and selects the action that has the maximum long-term cost-aware reward.

Action. Our algorithm sequentially analyzes regions with high zoom-in reward at high resolution. In this context, an *action* corresponds to selecting a region to analyze at high resolution. Each action a can be represented by a tuple (x, y, w, h) where (x, y) indicates the location, and (w, h) specifies the size of the region. At each step, the algorithm scores a set of potential actions—a list of rectangular regions—in terms of the potential long-term reward of taking those actions.

State. The representation encodes two types of information: 1) the predicted accuracy gain of regions yet to be analyzed; and 2) the history of regions that have already been analyzed at high resolution (the same region should not be zoomed in multiple times). We design a zoom-in accuracy gain regression network (R-net) to learn an informative accuracy gain map (AG map) as the state representation from which the zoom-in Q function can be successfully learned. The AG map has the same width and height as the input image. The value of each pixel in the AG map is an estimate of how much the detection accuracy might be improved if that pixel in the input image were included by the zoom-in region. As a result, the AG map provides detection accuracy gain for selecting different actions. After an action is taken, values corresponding to the selected region in the AG map decrease accordingly, so the AG map can dynamically record action history.

Cost-aware reward function. The state representation encodes the predicted accuracy gain of zooming in on each image subregion. To maintain a high accuracy with limited computation, we define a cost-aware reward function for actions. Given state s and action a, the cost-aware reward function scores each action (zoom region) by considering both cost increment and accuracy improvement as

$$R(s, a) = \sum_{k \ in \ a} |g_k - p_k^l| - |g_k - p_k^h| - \lambda \frac{b}{B} \tag{1}$$

where $k \ in \ a$ means that proposal k is included in the region selected by action a. p_k^l and p_k^h indicates coarse and fine detection scores, and g_k is the corresponding ground-truth label. The variable b represents the total number of pixels included in the selected region, and B indicates the total number of pixels of the input image. The first term measures the detection accuracy improvement. The second term indicates the zoom-in cost. The trade-off between accuracy and computation is controlled by the parameter λ. During training, the Q-net uses this reward function to calculate the immediate rewards of taking actions and learns a long-term reward function by Q learning [36].

3.2. Zoom-in accuracy gain regression network

The zoom-in accuracy gain regression network (R-net) predicts the accuracy gain of zooming in on a particular region based on the coarse detection results. The R-net is trained on pairs of coarse and fine detections so that it can observe how they correlate with each other to learn a suitable accuracy gain.

Toward this end, we apply two pre-trained detectors to a set of training images and obtain two sets of image detection results: low-resolution detections $\{(\mathbf{d}_i^l, p_i^l, \mathbf{f}_i^l)\}$ in the down-sampled image and high-resolution detections $\{(\mathbf{d}_j^h, p_j^h)\}$ in the high resolution version of each image, where $\mathbf{d}$ is the detection bounding box, p is the probability of being the target object and $\mathbf{f}$ indicates a feature vector of the corresponding detection. We use the superscripts h and l to indicate the high resolution and low resolution (down-sampled) images. For the model to learn whether or not a high resolution detection improves the overall results, given a set of coarse detections at training time, we introduce a *match layer* which associates detections produced by the two detectors. In this layer, we pair the coarse and fine detection proposals and generate a set of correspondences between them. The object proposals i in the down-sampled image and j in the high-resolution image are defined as corresponding to each other if we find a j with sufficiently large intersection over union IoU(d_i^l, d_j^h) with i (IoU >0.5).

Given a set of correspondences, $\{(\mathbf{d}_k^l, p_k^l, p_k^h, \mathbf{f}_k^l)\}$, we estimate the zoom-in accuracy gain of a coarse detection. A detector can handle only objects within a range of sizes, so applying the detector to the high-resolution image does not always produce the best accuracy. For example, larger objects might be detected with higher accuracy at lower resolution if the detector was trained on mostly smaller objects. So, we measure which detection (coarse or fine) is closer to groundtruth using the metric $|g_k - p_k^l| - |g_k - p_k^h|$ where $g_k \in \{0, 1\}$ indicates the groundtruth label. When the high resolution score p_k^h is closer to the groundtruth than the low resolution score p_k^l, the function indicates that this proposal is worth zooming in on. Otherwise, applying a detector on the down-sampled image is likely to yield a higher accuracy, so we should avoid zooming in on this proposal. We use a Correlation Regression (CR) layer to estimate the zoom-in accuracy gain of proposal k such that

$$\min_{\mathbf{W}} (|g_k - p_k^l| - |g_k - p_k^h| - \Phi(\mathbf{W}, f_k^l))^2 , \qquad (2)$$

where Φ represents the regression function and $\mathbf{W}$ indicates the parameters. The output of this layer is the estimated accuracy gain. The CR layer contains two fully connected layers where the first layer has 4,096 units and the second one has only one output unit.

The AG map can be generated given the learned accuracy gain of each proposal. We assume that each pixel inside a proposal bounding box has equal contribution to its accuracy gain. Consequently, the AG map is generated as

$$AG(x, y) = \begin{cases} \alpha \frac{\Phi(\widehat{\mathbf{W}}, f_k^l)}{b_k} & \textit{if } (x, y) \textit{ in } \mathbf{d}_k^l, \\ 0 & \textit{otherwise}, \end{cases} \qquad (3)$$

where (x, y) in $\mathbf{d}_k^l$ means point (x, y) is inside the bounding box $\mathbf{d}_k^l$ and b_k denotes the number of pixels included in $\mathbf{d}_k^l$. α is a constant number. $\widehat{\mathbf{W}}$ denotes the estimated parameters of the CR layer. The AG map is used as the state representation and it naturally contains the information of coarse detections' qualities. After zooming in and performing detection on a region, all the values inside the region are set 0 to prevent future zooming on the same region.

3.3. Zoom-in Q function learning network

The R-net provides information about which image region is likely to be the most informative if it is inspected next. Since the R-net is embedded within a sequential process, we use reinforcement learning to train a second network, the Q-net, to learn a long-term zoom-in reward function. At each step, the system takes an action by considering both immediate (Eq. 1) and future rewards. We formulate our problem in a Q learning framework, which approximates the long-term reward function for actions by learning a Q function. Based on the Bellman equation [5], the optimal Q function, $Q^*(s, a)$, obeys an important identity: given the current state, the optimal reward of taking an action equals the combination of its immediate reward and a discounted optimal reward at the next state triggered by this action (4)

$$Q^*(s, a) = \mathbb{E}_{s'}[R(s, a) + \gamma \max_{a'} Q^*(s', a')|s, a] \qquad (4)$$

where s is the state and a is an action. Following [29], we learn the Q function for candidate actions by minimizing the loss function at the i-th iteration, *i.e.*,

$$L_i = (R(s, a) + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2 \qquad (5)$$

where θ_i and θ_i^- are parameters of the Q network and those needed to calculate future reward at iteration i, respectively.

Eq. 5 implies that the optimal long-term reward can be learned iteratively if the immediate reward $R(s, a)$ is provided for a state-action pair. Since $R(s, a)$ is a cost-aware reward, the Q-net learns a long-term cost-aware reward function for the action set.

In practice, $\theta_i^- = \theta_{i-C}$ where C is a constant parameter. γ is future reward discount factor. We choose $C = 10$ and

$\gamma = 0.5$ empirically in our experiments. We also adopt the ϵ-greedy policy [34] at training to balance between exploration and exploitation. The ϵ setting is the same as in [7].

The structure of our Q-net is shown in Fig. 2. The input is the AG map and each pixel in the map measures the predicted accuracy gain if the pixel at that location in the input image is included in the zoom region. The output is a set of maps and each value of a map measures the long-term reward of taking the corresponding action (selecting a zoom region at a location with a specified size). To allow the Q-net to choose zoomed-in regions with different sizes, we use multiple pipelines, each of which outputs a map corresponding to zoomed-in regions of a specific size. These pipelines share the same features extracted from the state representation. In the training phase, actions from all maps are concatenated to produce a unified action set and trained end-to-end together by minimizing the loss function in Eq. 5 so that all actions values compete with each other.

After zooming in on a selected region, we get both coarse and fine detections on the region. We just replace the coarse detections with fine ones in each zoom-in region.

Window selection refinement. The output of the Q-net can be directly used as a zoom-in window. However, because candidate zoom windows are sparsely sampled, the window can be adjusted slightly to increase the expected reward. The Refine module takes the Q-net output as a coarse selection and locally moves the window towards a better location, as measured by the accuracy gain map by

$$\hat{a} = \arg\max_{a \in A} \sum_{(x,y)\ in\ a} AG(x, y) \qquad (6)$$

where $\hat{a}$ selects the refined window and $A = (x_q \pm \mu_x, y_q \pm \mu_y, w, h)$ corresponds to the local refinement area controlled by parameter μ, where (x_q, y_q, w, h) indicates the output window of Q-net. We show a qualitative example of refinement in Fig. 3.

4. Experiments

We perform experiments on the Caltech Pedestrian Detection dataset (CPD) [10] and a Web Pedestrian dataset (WP) collected from YFCC100M [21]. Datasets like Pascal VOC [12] and MS COCO [26] are not chosen to validate our method, because they are not close to our scenario. In [12] and [26], there are generally very few objects per image and most objects are large, which leads to 1) close-to-zero rewards for regions, since large objects are likely to maintain high detection accuracy after reasonable down sampling; and, 2) large zoom-in windows in order to enclose large objects. Low region rewards discourage the window selection process and large zoom-in windows produce high cost, which make our method invalid.

Caltech Pedestrian Detection (CPD). There are different settings according to different annotation types, *i.e.* *Overall*, *Near scale*, *Medium scale*, *No occlusion*, *Partial occlusion* and *Reasonable* [10]. Similar to the *Reasonable* setting, we only train and test on pedestrians at least 50 pixels tall. We sparsely sample images (every 30 frames) from the training set. There are 4,321 images in the training set and 4,088 images in the test set. We rescale the images to 600 pixels on the shorter side to form the high resolution version of image during both training and testing. All of our model components are trained on this training set.

Web Pedestrian (WP) dataset. The image resolution in the CPD dataset is low (640×480). To better demonstrate our approach, we collect 100 test images with much higher resolution from the YFCC100M [21] dataset. The images are collected by searching for keywords ”Pedestrian”, ”Campus” and ”Plaza”. An example is shown in Fig. 4 where pedestrians have varied sizes and are densely distributed in the images. For this dataset, we annotate all the pedestrians with at least 16-pixel width and less than 50% occlusion. Images are rescaled to 2,000 pixels on the longer side to fit for our GPU memory.

4.1. Baseline methods

We compare to the following baseline algorithms:

Fine-detection-all. This baseline directly applies the fine detector to the high resolution version of image. This method leads to high detection accuracy with high computational cost. All of the other approaches seek to maintain this detection accuracy with less computation.

Coarse-detection-all. This baseline applies the coarse detector on down-sampled images with no zooming.

GS+Rnet. Given the initial state representation generated by the R-net, we use a greedy search strategy (GS) to densely search for the best window every time based on the current state without considering the long-term reward.

ER+Qnet. The entropy of the detector output (object vs no object) is another way to measure the quality of a coarse detection. [2] used entropy to measure the quality of a region for a classification task. Higher entropy implies lower quality of a coarse detection. So, if we ignore the correlation between fine and coarse detections, the accuracy gain of a region can also be computed as

$$-p_i^l log(p_i^l) - (1 - p_i^l) log(1 - p_i^l) \qquad (7)$$

where p^l indicates the score of the coarse detection. For fair comparison, we fix all parameters of the pipeline except replacing the R-net output of a proposal with its entropy.

SSD and YOLOv2. We also compare our method with off-the-shelf SSD [27] and YOLOv2 [32] trained on CPD, to show the advantage of our method on large images.

4.2. Variants of our framework

We use *Qnet-CNN* to represent the Q-net developed using a fully convolutional network (see Fig. 2). To ana-

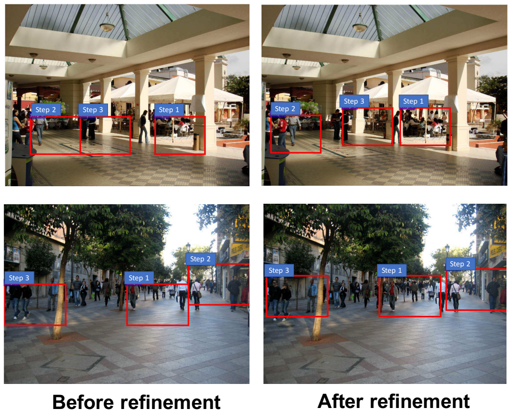

Figure 3: Effect of region refinement. Red boxes indicate zoom regions and the step number denotes the order that the zoom windows were selected. Before refinement, windows are likely to cut people in half due to the sampling grid, leading to a bad detection performance. Refinement locally adjusts the location of a window and produces better results.

lyze the contributions of different components to the performance gain, we evaluate three variants of our framework: Qnet*, Qnet-FC and Rnet*.

Qnet*. This method uses a Q-net with refinement to locally adjust the zoom-in window selected by Q-net.

Qnet-FC. Following [7], we develop this variant with two fully connected (FC) layers for Q-net. For *Qnet-FC*, the state representation is resized to a vector of length $1,200$ as the input. The first layer has 128 units and the second layer has 34 units (9+25). Each output unit represents a sampled window on an image. We uniformly sample 25 windows of size 320 × 240 and 9 windows of size 214 × 160 on the CPD dataset. Since the output number of *Qnet-FC* can not be changed, windows sizes are proportionally increased when *Qnet-FC* is applied to WP dataset.

Rnet*. This is an R-net learned using a reward function that does not explicitly encode cost ($\lambda = 0$ in Eq. 1).

4.3. Evaluation metric

We use three metrics when comparing to the *Fine-detection-all* strategy: AP percentage (A_{perc}), processed pixel numbers percentage (P_{perc}), and average detection time percentage (T_{perc}). A_{perc} quantifies the percentage of AP we obtain compared to the *Fine-detection-all* strategy. P_{perc} and T_{perc} indicate the computational cost as a percentage of the *Fine-detection-all baseline* strategy.

4.4. Implementation details

We downsample the high resolution image by a factor of 2 to form a down-sampled image for all of our experiments and only handle zoom-in regions at the high resolution.

For the Q-net, we spatially sample zoom-in candidate regions with two different window sizes (320 × 240 and 214 × 160) in a sliding window manner. For windows of size $W \times H$, we uniformly sample windows with horizontal stride $S_x = W/2$ and vertical stride $S_y = H/2$ pixels. For the refinement, we set $(\mu_x, \mu_y) = 0.5(S_x, S_y)$. The Q-net stops taking actions when the sum over all the values of the AG map is smaller than 0.1.

We use Faster R-CNN as our detector due to the success of R-CNN in many computer vision applications [15, 39, 24, 40, 19, 4]. Two Faster R-CNNs are trained on the CPD training set at the fine and coarse resolutions and used as black-box coarse and fine detectors afterwards. YOLOv2 and SSD are trained on the same training set with default parameter settings in the official codes released by the authors. All experiments are conducted using a K-80 GPU.

4.5. Qualitative results

The qualitative comparisons, which show the effect of refinement on the selected zoom-in regions, are shown in Fig. 3. We observe that refinement significantly reduces the cases in which pedestrians only partly occur in the selected windows. Due to the sparse window sampling of Q-net, optimal regions might not be covered by any window candidate, especially when the window size is relatively small compared to the image size.

We show a comparison between our method (*Q-net*-CNN+Rnet*) and the greedy strategy (*GS+Rnet*) in Fig. 4. *GS* tends to select duplicate zooms on the same portion of the image. While the Q-net might select a sub-optimal window in the near term, it leads to better overall performance in the long term. As shown in the first example of Fig. 4, this helps Q-net terminate with fewer zooms.

Fig. 5 shows a qualitative comparison of R-net and *ER*. The examples in the first row are detections that do not need to be zoomed in on, since the coarse detections are good enough. R-net produces much lower accuracy gains for these regions. On the other hand, R-net outputs much higher gains in the second row which includes regions needing analysis at higher resolution. The third row contains examples which get worse results at higher resolution. As we mentioned before, entropy cannot determine if zooming in will help, while R-net produces negative gains for these cases and avoids zooming in on these regions.

4.6. Quantitative evaluation

Table 1 shows the average precision (AP) and average detection time per image for *Fine-detection-all* and *Coarse-detection-all* strategies on CPD and WP datasets. The coarse baseline maintains only about 65% and 71% AP on CPD and WP, respectively, suggesting that the naive downsamping method significantly decreases detection accuracy.

Comparative results on the CPD and WP dataset are

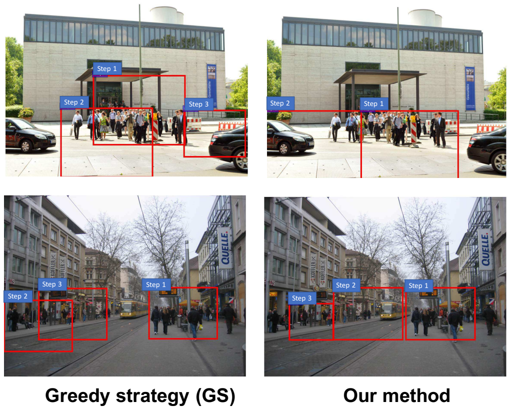

Figure 4: Qualitative comparison between using the Q-net* and a greedy strategy (GS) that selects the region with highest predicted accuracy gain at each step. Red bounding boxes indicate zoom-in windows and step number denotes the order of windows selection. The Q-net selects regions that appear sub-optimal in the near term but better zoom sequences in the long term, which leads to fewer steps as shown in the first row.

Dataset	AP$_f$	AP$_c$	DT$_f$(ms)	DT$_c$(ms)
CPD	0.493	0.322	304	123
WP	0.407	0.289	1375	427

Table 1: *Coarse-detection-all*(with subscript c) v.s. *Fine-detection-all* (with subscript f) on CPD and WP datasets. DT indicates average detection time per image.

shown in Table 2. *Q-net*-CNN* + *R-net* reduces processed pixels by over 50% with comparable (or even better) detection accuracy than the *Fine-detection-all* strategy and improves detection accuracy of *Coarse-detection-all* by about 35% on the CPD dataset. On the WP dataset, the best variant (*Q-net*-CNN* + *R-net**)) reduces processed pixels by over 60% while maintaining 97% detection accuracy of *Fine-detection-all*. Table 2 shows that variants of our framework outperform *GS+Rnet* and *Qnet+ER* in most cases which suggests that *Qnet* and *Rnet* are better than *GS* and *ER*. Q-net is better than *GS* since the greedy strategy considers individual actions separately, while Q-net utilizes a RL framework to maximize the long term reward.

Qnet-CNN+Rnet* always produces better detection accuracy than *Qnet*-CNN+ER* under the same cost budget, which demonstrates that learning the accuracy gain using an R-net is preferable to using entropy, a hand-crafted measure. This could be due to two reasons: 1) entropy measures only the confidence of the coarse detector, while our R-net estimates the correlation with the high-resolution detector based on confidence and appearance; 2) according to the regression target function in Eq. 2, our R-net also measures whether the zoom-in process will improve detection accuracy. This avoids wasting resources on regions that cannot be improved (or might even be degraded) by fine detections.

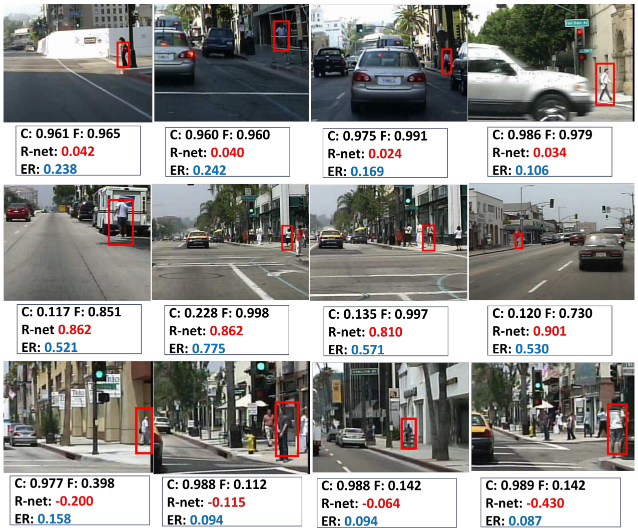

Figure 5: Qualitative comparison of R-net and ER on the Caltech Pedestrians test set. The first row of numbers indicate probability of the red box being a pedestrian. C denotes coarse detection and F indicates fine detection. Red font denotes the accuracy gain of R-net and blue is for ER. Positive and negative values are normalized to [0, 1] and [-1, 0). Compared to ER, R-net gives lower positive scores (row #1)/ negative scores (row #3) for regions that coarse detections are good enough/ better than fine detections and it produces higher scores for regions (row #2) where fine detections are much better than coarse ones.

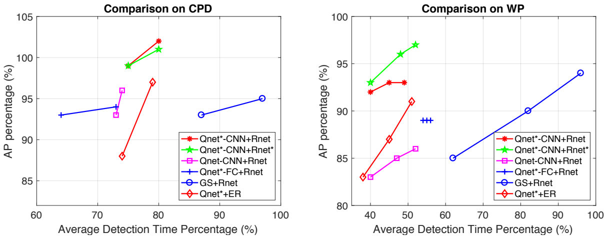

Figure 6: Detection time and accuracy comparison on the CPD/WP dataset after zooming in on two/three regions.

We observe from Fig. 6 that our approach (*Qnet*-CNN+Rnet* and *Qnet*-CNN+Rnet**) reduces detection time by 50% while maintaining a high accuracy on the WP dataset. On the CPD dataset, they can reduce detection time by 25% without a significant drop of accuracy. Detection time cannot be reduced as much as on the WP dataset, since CPD images are relatively small; however, it is notable that our approach helps even in this case.

		Baselines		**Variants under our framework**			
	P_{perc}	GS+Rnet	Qnet*-CNN+ER	Qnet*-CNN+Rnet	Qnet*-CNN+Rnet*	Qnet*-FC+Rnet	Qnet-CNN+Rnet
CPD	≤40%	65%(40%)	88%(74%)	**99%**(75%)	65%(40%)	93%(64%)	65%(40%)
	≤45%	93%(87%)	97%(79%)	**102%**(80%)	101%(80%)	94%(73%)	96%(73%)
	≤50%	95%(97%)	97%(79%)	**102%**(80%)	101%(80%)	94%(73%)	96%(73%)
WP	≤30%	85%(62%)	83%(38%)	92%(40%)	**93%**(40%)	71%(31%)	83%(40%)
	≤35%	90%(82%)	91%(51%)	93%(45%)	**96%**(48%)	71%(31%)	85%(47%)
	≤40%	94%(96%)	91%(51%)	93%(49%)	**97%**(52%)	89%(54%)	86%(52%)

Table 2: Detection accuracy comparisons in terms of A_{perc} on the CPD and WP datasets under a fixed range of processed pixel percentage (P_{perc}). Bold font indicates the best result. Numbers are display as $A_{perc}(T_{perc})$- T_{perc} is included in the parentheses for the reference of running time. Note that 25% P_{perc} overhead is incurred simply by analyzing the downsampled image (this overhead is included in the table) and percentages are relative to *Fine-detection-all* baseline (an A_{perc} of 80% means that an approach reached 80% of the AP reached by the baseline).

	CPD		**WP**	
	AP	DT(ms)	AP	DT(ms)
SSD500 [27]	0.405	128	0.255	570
SSD300 [27]	0.400	74	0.264	530
YOLOv2 [32]	0.398	70	0.261	790
Our method	**0.503**	243	**0.379**	619

Table 3: Comparison between Qnet*-CNN+Rnet and single-shot detectors trained on CPD. DT indicates average detection time per image. Bold font indicates the best result.

Table 3 shows accuracy/cost comparisons between YOLO/SSD and our method. Experiments suggest the following conclusions: 1) although fast, these single-shot detectors achieve much lower AP on images with objects occurring over a large range of scales; 2) as image size increases, YOLO/SSD processing time increases dramatically, while, our method achieves much higher accuracy with comparable detection time; 3) SSD consumes much more GPU memory than other detectors on large images due to the heavy convolution operations. We have to resize images of WP to 800×800 to fit within GPU memory. Note that it is possible to improve the results of YOLO/SSD by pruning the networks or training with more data, but that is not within the scope of this paper.

4.7. Ablation analysis

Improvement by refinement (*Qnet*-CNN+Rnet* vs. *Qnet-CNN+Rnet*). In Table 2, we find that region refinement significantly improves detection accuracy under fixed cost ranges, especially on the WP. Refinement is more useful when zoom-in window size is relatively small compared with image size due to the sparse window sampling of Q-net. Fig. 3 qualitatively shows the effect of refinement.

Improvement by CNN (*Qnet*-CNN+Rnet* vs. *Qnet*-FC+Rnet*). FC has two obvious drawbacks in our setting. First, it has a fixed number of inputs and outputs which makes it hard to handle images with different sizes. Second, it is spatially dependent. Images from the CPD dataset consist of driving views which have strong spatial priors, *i.e.*, most pedestrians are on the sides of the street and the horizon is roughly in the same place. *Qnet-FC* takes advantage of these spatial priors, so it works better on this dataset. However, when it is applied to the WP dataset, its performance drops significantly compared to other methods, since the learned spatial priors now distract the detector.

Improvement by the cost term (*Qnet*-CNN+Rnet* vs. *Qnet*-CNN+Rnet).** *Qnet*-CNN+Rnet* outperforms *Qnet*-CNN+R-net** on CPD, especially when P_{perc} is low (40%). Without explicit cost penalization, the algorithm often selects the largest zoom regions, a poor strategy when there is a low pixel budget. However, since the window sizes are relatively small compared to the image size of the WP dataset, *Qnet*-CNN+Rnet** does not suffer much from this limitation. On the contrary, it benefits from zooming in on relatively bigger regions. Consequently, it outperforms other variants. Nevertheless, *Qnet*-CNN+Rnet* has comparable detection accuracy and can generalize better on scenarios where window sizes are comparable with image size.

5. Conclusion

We propose a dynamic zoom-in network to speed up object detection in large images without manipulating the underlying detector's structure. Images are first downsampled and processed by the R-net to predict the accuracy gain of zooming in on a region. Then, the Q-net sequentially selects regions with high zoom-in reward to conduct fine detection. The experiments show that our method is effective on both Caltech Pedestrian Detection dataset and a high resolution pedestrian dataset.

Acknowledgement. This work was partially supported by the Defense Advanced Research Projects Agency (DARPA) under contract no. W911NF-16-C-0003. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of DARPA or the U.S. Government.

References

[1] B. Alexe, N. Heess, Y. W. Teh, and V. Ferrari. Searching for objects driven by context. In *Advances in Neural Information Processing Systems*, pages 881–889, 2012.

[2] A. Almahairi, N. Ballas, T. Cooijmans, Y. Zheng, H. Larochelle, and A. Courville. Dynamic capacity networks. In *International Conference on Machine Learning*, pages 2549–2558, 2016.

[3] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[4] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *CoRR*, abs/1512.04143, 2015.

[5] R. Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10):767–769, 1956.

[6] R. Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[7] J. C. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015.

[8] X. S. Chen, H. He, and L. S. Davis. Object detection in 20 questions. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016.

[9] E. L. Denton, W. Zaremba, J. Bruna, Y. Lecun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems 27*, pages 1269–1277. Curran Associates, Inc., 2014.

[10] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2012.

[11] I. Endres and D. Hoiem. Category independent object proposals. In *European Conference on Computer Vision*, pages 575–588. Springer, 2010.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[14] M. Figurnov, D. P. Vetrov, and P. Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions. *CoRR*, abs/1504.08362, 2015.

[15] M. Gao, A. Li, R. Yu, V. I. Morariu, and L. S. Davis. C-wsl: Count-guided weakly supervised localization. *arXiv preprint arXiv:1711.05282*, 2017.

[16] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[18] H. He, H. Daumé III, and J. Eisner. Cost-sensitive dynamic feature selection. In *ICML Inferning Workshop*, 2012.

[19] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[20] Z. Jie, X. Liang, J. Feng, X. Jin, W. Lu, and S. Yan. Tree-structured reinforcement learning for sequential object localization. In *Advances in Neural Information Processing Systems*, pages 127–135, 2016.

[21] S. Kalkowski, C. Schulze, A. Dengel, and D. Borth. Real-time analysis and visualization of the yfcc100m dataset. In *Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions*, pages 25–30. ACM, 2015.

[22] S. Karayev, M. Fritz, and T. Darrell. Anytime recognition of objects and scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 572–579, 2014.

[23] Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *CoRR*, abs/1511.06530, 2015.

[24] A. Li, J. Sun, J. Y.-H. Ng, R. Yu, V. I. Morariu, and L. S. Davis. Generating holistic 3d scene abstractions for text-based image retrieval. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[25] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[28] Y. Lu, T. Javidi, and S. Lazebnik. Adaptive object detection using adjacency and zoom prediction. *arXiv preprint arXiv:1512.07711*, 2015.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[30] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898, 2014.

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[32] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[33] 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*, pages 91–99, 2015.

[34] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[35] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[36] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[37] R. Yu, X. Chen, V. I. Morariu, and L. S. Davis. The role of context selection in object detection. In *British Machine Vision Conference (BMVC)*, 2016.

[38] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis. Nisp: Pruning networks using neuron importance score propagation. *arXiv preprint arXiv:1711.05908*, 2017.

[39] R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation. *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[40] R. Yu, H. Wang, and L. S. Davis. Remotenet: Efficient relevant motion event detection for large-scale home surveillance videos. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[41] X. Zhang, J. Zou, X. Ming, K. He, and J. Sun. Efficient and accurate approximations of nonlinear convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[42] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 249–258, 2015.

[43] Z. Zhang, Y. Liu, T. Bolukbasi, M.-M. Cheng, and V. Saligrama. Bing++: A fast high quality object proposal generator at 100fps. *arXiv preprint arXiv:1511.04511*, 2015.

[44] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014.