Efficient Object Detection in Large Images Using Deep Reinforcement Learning

Burak Uzkent Christopher Yeh Stefano Ermon
Department of Computer Science, Stanford University
buzkent@cs.stanford.edu,chrisyeh@stanford.edu,ermon@cs.stanford.edu

Abstract

Traditionally, an object detector is applied to every part of the scene of interest, and its accuracy and computational cost increases with higher resolution images. However, in some application domains such as remote sensing, purchasing high spatial resolution images is expensive. To reduce the large computational and monetary cost associated with using high spatial resolution images, we propose a reinforcement learning agent that adaptively selects the spatial resolution of each image that is provided to the detector. In particular, we train the agent in a dual reward setting to choose low spatial resolution images to be run through a coarse level detector when the image is dominated by large objects, and high spatial resolution images to be run through a fine level detector when it is dominated by small objects. This reduces the dependency on high spatial resolution images for building a robust detector and increases run-time efficiency. We perform experiments on the xView dataset, consisting of large images, where we increase run-time efficiency by 50% *and use high resolution images only* 30% *of the time while maintaining similar accuracy as a detector that uses only high resolution images.*

1. Introduction

Deep Convolutional Neural Networks (CNNs) have been successfully applied to different computer vision tasks including image recognition [20, 13, 42], object detection [26, 35, 33], object tracking [19, 2, 48, 49]. Traditionally, CNNs use images resized to a pre-determined number of pixels for each dimension. For instance, CNNs for image recognition typically use images resized to 224×224px to utilize weights pre-trained on ImageNet [15, 14].

On the other hand, convolutional object detectors are traditionally applied to images resized to dimensions less than 1,000 pixels on each side (e.g.∼500px for images from the MSCOCO and PASCAL VOC2007/VOC2012 datasets [27], and ∼600px for satellite images from the xView dataset [39]). However, in some application domains, images may be much larger, on the order of several thousands of pixels in each dimension. These large images enable higher detection accuracy, especially for smaller objects relative to the field of view [33], but they require additional computation, time, and/or financial resources to process.

For example, in the field of self-driving vehicles or traffic monitoring, fusing together images from multiple sensors produces large images, which presents a computational challenge to maintain real-time processing [54, 30]. Similarly, most satellite images in the xView dataset have about 1,000-10,000px on each side [22]. Yet, in realistic remote sensing applications, these high spatial resolution satellite images (e.g.<1m/px) are financially costly to acquire, compared to publicly available low spatial resolution satellite images (e.g. 10m/px) [6]. Therefore, it is desirable to build a system that minimizes dependency on large images to reduce the costs of analyzing satellite images for computer vision tasks including image recognition [41, 50], object detection [22, 4], object tracking [46, 47], and poverty mapping [40].

Directly applying existing state-of-the-art convolutional detectors on large images not only increases processing time but also the memory required to store large feature maps [23, 52]. The traditional *sliding window* approach mitigates the memory requirement by "sliding" the convolutional detector over crops of the image until the full image has been processed. However, the processing time of this technique increases quadratically with respect to side-length of the image, and it may have lower accuracy because the detector is unable to "see" the whole image at once.

Considering the trade-off between *accuracy* and various *costs* associated with using high spatial resolution images, we propose an adaptive framework that only chooses high spatial resolution images when fine information is required, and uses low spatial resolution images when the coarse information provides sufficient information for the objects of interest. This maintains the overall detection accuracy while increasing run-time efficiency and reducing the dependency on expensive, high resolution (HR) imagery.

To describe our approach at a high level, we train an agent using *reinforcement learning* to explicitly maximize accuracy while minimizing the amount of HR images. It

first processes a low resolution (LR) image, learning the global image context, then selectively requests certain HR patches that it deems necessary for making accurate object detections. Importantly, the agent is not given the HR image, thus limiting potential acquisition costs for the HR image to only the patches that it selects. Experiments on the xView dataset show that compared to an object detector that exclusively uses HR images, our method maintains nearly the same accuracy but uses HR images only about 30% of the time, in turn increasing run-time performance by about 50%.

2. Related Work

Convolutional Object Detection Earlier convolutional object detectors such as R-CNN and Fast-RCNN [9, 8] use a selective search algorithm [45] to identify box proposals to run the detector on. On the other hand, Faster-RCNN [36] jointly learns box proposals and the classifier in an end-to-end manner [36], but the two-stage design limits computational efficiency. Instead, single-shot detectors [27, 34, 25] directly predict bounding boxes from fixed anchors on input images without relying on a region proposal network, thereby achieving real-time performance. To replace fixed anchors with sparse adaptive anchors, [28] propose a search method to predict anchor boxes that are likely to contain objects. Their method improves efficiency on cases with sparse object instances.

Sliding Windows on Large Images Running single and two stage detectors on large images requires a large amount of memory to store large feature maps [33, 7]. To avoid that, the traditional sliding window technique divides the image into smaller (often overlapping) windows and then runs the detector on each window [11, 44, 52, 23]. While this approach has the same memory footprint as a detector running on a single window, the number of windows (and therefore run time) increases quadratically in the image side length. Furthermore, this technique is wasteful if the detector is run on a window that contains no objects or has objects large enough to be detected in a LR image.

Pruning the image search space Various techniques have been proposed to reduce the search space, such as selectively choosing windows to evaluate based on previously observed windows [1, 10, 31]. However, such systems introduce non-trivial overhead on the order of seconds per image, and they are not designed to take advantage of the global image context. Other works adopt a cascade of object detectors to narrow the search space: [24] starts with downsampling the sliding windows to 12×12px images to process with a shallow 12Net to easily remove negatives. They then process the faces detected by 12Net using the 24Net with 24×24px images. On the other hand, [21] adopts a two-stage CNN where the first stage learns a rough heat map of moving vehicles whereas the second stage processes the chosen ROIs to localize the vehicles. However, both [24, 21] assume access to the HR image corresponding to every object in the image, which may be costly to obtain, and [21] additionally uses temporally consecutive video frames of the same region, which are not available everywhere and only help locate non-stationary objects.

Reinforcement Learning for Efficient Detection Reinforcement Learning (RL) has been recently used to (1) *replace classical detectors such as SSD and Faster-RCNN*, (2) *replace exhaustive box proposal techniques in two-stage detectors*, and (3) *find ROIs in very large images to run a detector on*. Most of the methods proposed in this categories focus on learning *sequential policies*. Under category (1), [3, 29] proposed a top-down sequential object detection models trained with the Q-learning algorithm. [3] uses an action space that deforms the bounding box by applying translation and scale factors whereas [29] uses actions to fixate a bounding box in image space. Most of the RL methods associated with object detection fall into category (2). For example, [16] recursively divides up an image in a top-down approach where the divisions are decided by the RL agent. The box proposals returned by the agent are then passed through Fast-RCNN. Some other studies use RL for sequentially finding box proposals to replace the first stage of two-stage detectors [10, 32]. Our approach, like [7], falls into category (3) where a RL agent is trained to examine a down-sampled image and sequentially choose ROIs to zoom-in on. For efficiency, [7] directly use the detections on the full down-sampled image if the improvement gained by zooming-in to a ROI is not sufficiently high. Unlike [16], they use a *cost-sensitive* reward function to limit number the of steps as in [12, 17]. Similarly, we learn the zoom-in policies with a *cost-sensitive* reward function. On the other hand, they downsample the initial large image by a factor of *2* and focus on pedestrians represented by reasonable number of pixels. In contrast, we consider a higher downsampling ratio for the policy network on very large remote sensing images (e.g. $>$1,000px on each side) in which objects are represented by a much *smaller number of pixels*. Additionally, similar to our study, their zoom-in ROIs can contain *multiple instances* of objects whereas [10, 29, 32, 16] search for the boxes surrounding a *single object*. Finally, all of the previous methods proposed for efficient detection learn sequential policies whereas ours chooses the zoom-in ROIs in *single forward pass*. This is beneficial for *parallelizing* object detection on zoom-in ROIs [52].

3. Proposed Formulation

We propose an efficient object detection framework consisting of two modules named *coarse* and *fine* level search. With coarse level search, we perform an initial search space optimization on *very large* images with over 3,000 pixels in each dimension. At the end of coarse level search, we

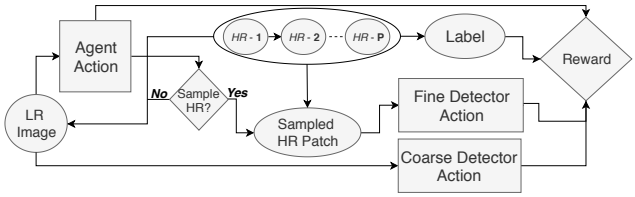

Figure 1. Proposed Bayesian Decision *influence* diagram. At training time, HR images are downsampled to LR images which the agent uses for sampling actions. Detector (coarse/fine) then uses LR/HR image to output bounding boxes which are used to compute reward together with the labels from HR images.

find an initial set of patches where it may be beneficial to zoom-in and acquire HR images. In fine level search, we perform further search space optimization on the patches chosen by the coarse level search module to make a final decision about which subpatches to acquire HR images for. Both coarse and fine level search modules are formulated similarly with a two-step episodic Markov Decision Process (MDP), as shown in our generic influence diagram in Fig. 1. In the diagram, we represent random variables with a circle, actions with a square/rectangle, and utilities with a diamond.

Coarse Level Search This first module of our framework, implemented as *CPNet*, finds ROIs/patches to zoom into, conditioned on the low spatial resolution image of the initial large area. This is achieved by applying the proposed generic influence diagram as shown in Fig. 1 to coarse level search. In this direction, a large HR image $x_H = (x_H^1, x_H^2, \ldots, x_H^{P_c})$ is composed of equal-size non-overlapping patches, where P_c is the number of coarse-level patches. Unlike in traditional computer vision settings, x_H is latent, *i.e.*, it is *not observed by the agent*. $Y = \{Y_1, \ldots, Y_{P_c}\}$ is an array of arrays of the (unobserved) ground truth bounding boxes associated with each patch of x_H, where $Y_i = \{y_1, \ldots, y_{P_f}\}$. Each bounding box is represented as a tuple $y_i^j = (g_x, g_y, w, h, c)$, which is a random variable containing the centroid, width, height, and object class. The random variable $x_L = (x_L^1, x_L^2, \ldots, x_L^{P_c})$ denotes the LR image of the same scene as x_H, where x_L^i represents the lower spatial resolution version of x_H^i.

In the first step of the MDP for coarse level search, the agent observes x_L and outputs a binary action array, $a_c \in \{0,1\}^{P_c}$, where $a_c^i = 1$ means that the agent would like to consider acquiring HR subpatches of the i-th patch x_H^i. We define the patch sampling policy model, parameterized by θ_p^c, as

$$\pi_c(a_c|x_L;\theta_p^c) = p(a_c|x_L;\theta_p^c) \tag{1}$$

where $\pi_c(x_L;\theta_p^c)$ is a function mapping the observed LR image to a probability distribution over patch sampling actions a_c. The joint probability distribution over the random variables x_H, Y, x_L, and action a_c, can be written as

$$\begin{aligned} &p(x_H, x_L, Y, a_c) \\ &= p(x_H)\, p(Y|x_H)\, p(x_L|x_H)\, p(a_c|x_L;\theta_p^c). \end{aligned} \tag{2}$$

In the second step of the MDP, the agent runs the object detection policy. Conditioned on a_c, it observes either x_H^i or x_L^i and chooses an action $a_d = \hat{Y}_i$ where $\hat{Y}_i = \{\hat{y}_1 \cdots \hat{y}_{P_f}\}$ and $\hat{y}_i^j = (\hat{g}_x, \hat{g}_y, \hat{w}, \hat{h}, \hat{c})$ represents a predicted bounding box for x_H^i or x_L^i. We define the object detection policy as follows:

$$\pi_d(a_d|x_L^i;\theta_d^c) = p(a_d|x_L^i;\theta_d^c), \tag{3}$$

$$\pi_d(a_d|x_H^i;\theta_d^f) = p(a_d|x_H^i;\theta_d^f), \tag{4}$$

where θ_d^c and θ_d^f represent the coarse and fine object detectors operating on x_L^i and x_H^i.

The overall objective J_c is defined to maximize the expected utility R_c given the evidence, represented by

$$\max_{\theta_p^c, \theta_d^f, \theta_d^c} J_c(\theta_p^c, \theta_d^f, \theta_d^c) = \mathbb{E}_p[R_c(a_c, a_d, Y)], \tag{5}$$

where the utility depends on a_c, a_d, and Y. The reward penalizes the agent for selecting a large number of HR patches (e.g., based on the norm of a_c) and includes a performance metric evaluating the accuracy of the detector, a_d, given the true label Y (e.g., recall, precision). We detail the reward function in Section 4.3.

Fine Level Search The CPNet alone can be satisfactory when considering images with <1,000px on each side. However, for very large images (e.g. >1,000px on each side), it is desirable to further optimize the search space as a *patch can be represented by a large number of pixels*. In such images, one way to further optimize the search space is to reduce the size of patches for CPNet, resulting in a larger action space. However, training CPNet with a larger action space can be unstable and take a drastically longer time. For this reason, we use another policy network, called *FPNet*, parameterized by θ_p^f, and apply it to the random variable low spatial resolution patch x_L^i sampled from the latent variable $x_H^i = (x_h^1, x_h^2, \ldots, x_h^{P_f})$ consisting of P_f overlapping fine-level subpatches. The task of this policy network is to choose a binary action array $a_f \in \{0,1\}^{P_f}$, where $a_f^j = 1$ means that the agent would like to acquire the j-th HR subpatch x_h^j. The subpatch sampling policy model is parameterized by θ_p^f and formulated similarly to Eq 1.

The first step of the MDP can then be modeled similarly to the CPNet with a joint probability distribution over the random variables as

$$\begin{aligned} p(x_H^i, x_L^i, Y^i, a_f) = p(x_H^i)p(Y^i|x_H^i) \\ p(x_L^i|x_H^i)p(a_f|x_L^i;\theta_p^f). \end{aligned} \tag{6}$$

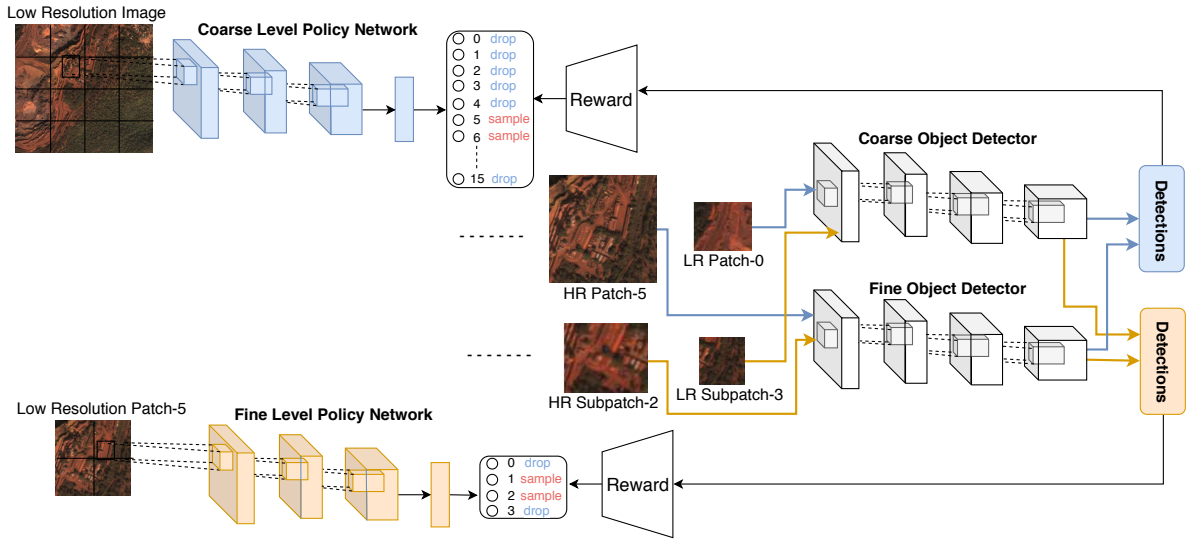

Figure 2. The proposed coarse and fine level policy networks (CPNet, and FPNet) to process *very large images* (e.g. >1,000px on each side). The CPNet uses the initial large LR image to choose a set of actions representing a unique patch in the image space. The sampled patch is then either used by coarse or fine detector to estimate the expected reward. The FPNet uses a patch of an initial large LR image to choose a set of actions. Next, the coarse or fine detector is run on the sampled subpatch. We train CPNet and FPNet independently. In test time, in our cascaded approach, we first run CPNet on large image and run FPNet on patches asked to be sampled by CPNet.

In the second step of the MDP, the agent observes the random variables, x_h^j or x_l^j for $j = \{1, \dots, P_f\}$, based on a_f and chooses an action $a_d = \hat{Y}_i$, where $\hat{Y}_i = \{\hat{y}_1, \dots, \hat{y}_{P_f}\}$ and $\hat{y}_i^m = (\hat{g}_x, \hat{g}_y, \hat{w}, \hat{h}, \hat{c})$ represents a predicted bounding box for x_h^j or x_l^j. We then define the object detection policy similarly to Eq 3 and 4. Finally, the overall objective function J_f is defined as maximizing the expected utility R_f given the evidence, represented by

$$\max_{\theta_p^f, \theta_d^f, \theta_d^c} J_f(\theta_p^f, \theta_d^f, \theta_d^c) = \mathbb{E}_p[R_f(a_f, a_d, Y_i)]. \quad (7)$$

Fig. 2 visualizes the proposed CPNet and FPNet in detail.

4. Proposed Solution

4.1. Modeling the Policy Networks and Detectors

In the previous section, we formulated the task of efficient object detection in a cascaded approach where coarse and fine level search is formulated as a two step episodic MDP. Here, we detail the action space and how the policy distributions for a_c, a_f and a_d are modelled for each module. To represent our discrete action space for a_c and a_f, we divide the image space into equal size patches, resulting in P_c and P_f number of patches and subpatches. In this study, for the coarse and fine level search, we use $P_c = 16$ and $P_f = 4$ regardless of the size of the input image and leave the task of choosing variable size bounding boxes as a future work. In the first step of the two step MDP, the policy networks, f_p^c and f_p^f, *output the probabilities for all the actions at once* after observing x_L and x_L^i. An alternative approach could be in the form of a framework where $a_{c,f}^{i,j}$ is conditioned on $a_{c,f}^{1:i-1,1:j-1}$. However, the proposed concept of *outputting all the actions at once* provides a more efficient decision making process.

In this study, we model the action likelihood function of the policy networks, f_p^c and f_p^f, by multiplying the probabilities of the individual HR patch/subpatch selections, represented by Bernoulli distributions as follows:

$$\pi_c(a_c|x_L, \theta_p^c) = \prod_{i=1}^{P_c} s_c^i (1 - s_c^i)^{(1-a_c^i)}, \quad (8)$$

$$\pi_f(a_f|x_L^i, \theta_p^f) = \prod_{j=1}^{P_f} s_f^j (1 - s_f^j)^{(1-a_f^j)} \quad (9)$$

where s is the prediction vector formulated as

$$s_c = f_p^c(x_L; \theta_p^c), \quad (10)$$

$$s_f = f_p^f(x_L^i; \theta_p^f). \quad (11)$$

To get probabilistic values, $s_c, s_f \in [0, 1]$, we use a sigmoid function on the final layers of CPNet and FPNet.

The next set of actions for the coarse level search, a_d, is chosen by the coarse and fine level object detectors using the LR patch x_L^i or the sampled HR patch x_H^i. For the fine level search, the actions, a_d, are chosen using the LR subpatch x_l^i or the sampled HR subpatch x_h^i.

4.2. Training the Policy Networks

After defining the two step MDP and modeling the policy and detector networks, we detail the training procedure of the proposed efficient object detection model. The goal of training is to learn the optimal parameters of policy networks, θ_p^c and θ_p^f. Because the actions are discrete, we cannot use the reparameterization trick to optimize the objective w.r.t. θ_p^c and θ_p^f. To optimize the parameters θ_p^c, θ_p^f of f_p^c and f_p^f, we need to use model-free reinforcement learning algorithms such as Q-learning [51] and policy gradient [43]. Policy gradient is more suitable in our scenario since the number of unique actions the policy network can choose is 2^P and increases exponentially with P. Finally, we use the *REINFORCE* method [43] to optimize the objective w.r.t. policy network parameters as:

$$\nabla_{\theta_p^c} J_c = \mathbb{E}\left[R_c(a_c, a_d, Y)\nabla_{\theta_p^c} \log \pi_{\theta_p^c}(a_c|x_L)\right], \quad (12)$$

$$\nabla_{\theta_p^f} J_f = \mathbb{E}\left[R_f(a_f, a_d, Y_i)\nabla_{\theta_p^f} \log \pi_{\theta_p^f}(a_f|x_L^i)\right]. \quad (13)$$

Averaging across a mini-batch via Monte-Carlo sampling produces an unbiased estimate of the expected value, but with potentially large variance. Since this can lead to an unstable training process [53], we replace $R_c(a_c, a_d, Y)$ in Eq. 12 with the advantage function to reduce the variance:

$$\nabla_{\theta_p^c} J_c = \mathbb{E}\left[A \sum_{i=1}^{P_c} \nabla_{\theta_p^c} \log(s_c^i a_c^i + (1 - s_c^i)(1 - a_c^i))\right] \quad (14)$$

$$A(a_c, \hat{a}_c, a_d, \hat{a}_d) = R_c(a_c, a_d, Y) - R_c(\hat{a}_c, \hat{a}_d, Y) \quad (15)$$

where $\hat{a}_c$ and $\hat{a}_d$ represent the baseline action vectors. To get $\hat{a}_c$, we use the most likely action vector proposed by the policy network: *i.e.*, $a_c^i = 1$ if $s_c^i > 0.5$ and $a_c^i = 0$ otherwise. The coarse and fine level detectors, f_d^c and f_d^f, then observes x_L^i or x_H^i, and outputs the predicted bounding boxes $\hat{a}_d$. The advantage function assigns the policy network *a positive value* only when the action vector sampled from Eq. 9 produces higher reward than the action vector with maximum likelihood, which is known as a self-critical baseline [37]. Similarly to the coarse level search, we introduce the advantage function to the *fine level search* module and do not show it in this section for simplicity.

Finally, in this study we use temperature scaling [43] to encourage exploration during training time by bounding the probabilities of the policy networks as

$$s = \alpha s + (1 - \alpha)(1 - s). \quad (16)$$

In our experiments, we tune α to $0 < \alpha < 1$, where we observe that setting it to a small value produces more uniform probabilities to sample off-policy actions.

4.3. Modeling the Reward Function

The proposed framework uses the policy gradient reinforcement learning algorithm to learn the parameters of the policy networks, adjusting their weights to increase the expected reward value. Thus, it is crucial to design a reward function reflecting the desired characteristics of an efficient object detection method for large images: low *image acquisition cost* and high *run-time efficiency*. For the coarse level policy network, our reward function R_c encourages the use of LR image patches x_L^i with the coarse level detector f_d^c. We define R_c as follows, where $\hat{Y}^f$ are the object detections by the fine-level object detector on patches of x_H, and $\hat{Y}^c$ are the detections by the coarse-level detector on x_L:

$$R_c = R_{acc}(\hat{Y}^f, \hat{Y}^c, Y) + R_{cost}(a_c) \quad (17)$$

$$R_{acc} = \sum_{i=1}^{P_c} (Recall(\hat{Y}_i^f, Y_i) - (Recall(\hat{Y}_i^c, Y_i) + \beta)) \cdot N_i \quad (18)$$

$$R_{cost} = (\sigma + \lambda)(1 - |a_c|_1)/P_c \quad (19)$$

where R_{acc} is detection recall and R_{cost} combines image acquisition cost and run-time performance reward. The R_{acc} term encourages zooming-in when the recall difference between the coarse and fine detector is positive. The difference is then scaled with the number of objects in the patch, N_i, to prioritize zooming-in to regions where *there are more objects*. β, on the other hand, prioritizes using LR images when the recall values from coarse and fine detector are similar. Note that since our priority is to *minimize the ratio of false negatives*, we only use Recall and do not consider *Precision* in R_{acc}. The other component of reward, R_{cost}, represents the run-time and image acquisition costs with their own coefficients λ and σ. In this case, the reward increases *linearly* with the smaller number of zoomed-in patches. The reward function R_f for the fine-level policy network is defined similarly.

5. Experiments

5.1. Baselines and State-Of-The-Art Models

Sliding Window A simple method for running object detectors on large images is the *sliding window* approach. Using this fixed policy approach, we either apply a coarse level detector, f_d^c, on x_l^j or a fine level detector, f_d^f, on x_h^j for $j = 1, \ldots, P_f$. We then repeat this for every x_H^i or x_L^i for $i = 1, \ldots, P_c$.

Random Policy In this case, we sample patch specific probabilities from a policy represented by patch specific uniform distribution.

Entropy Based Policy Sampling In this case, we first the compute the patch-wise confidence of the coarse level object detector, $s_c^i = \frac{1}{M}\sum_{m=1}^{M} c_m$, on x_L^i where M represents the number of detected bounding boxes. Next, we

pass the patch to the fine level search if s_c^i is larger than a threshold. In the fine level search, we compute s_f^j similarly to the coarse level search given that $a_c^i = 1$ and acquire a HR subpatch x_h^j if s_f^j is larger than a threshold and use it for fine level object detection. Otherwise, we use x_l^j for coarse level object detection.

Dynamic Zoom-in Network [7] proposed the *state-of-the-art* method for efficiently detecting objects in large images without changing the underlying structure of the detector. They show the results on *640x480* pixels images from the Caltech Pedestrian Detection dataset [5]. Their method is not suitable for larger images, i.e. ∼3,000px, as it starts with a detector trained on *full images* downsampled by 2. To make it practical for larger images, we train a coarse detector on the full images downsampled by a larger scale.

Variants of the Proposed Approach Additionally, in this section, we report the *coarse level only* and *fine level only* results for both the baseline models and proposed approach. In the coarse level only method, we run the coarse level policy network on x_L and acquire HR image x_H^i if $a_c^i = 1$ and use it for fine level object detection. In the fine level only approach, we run the fine level policy network on x_L^i unconditioned to a_c. Then, we acquire a HR image x_h^j if $a_f^j = 1$ and use it for fine level object detection.

5.2. Implementation Details

Policy Networks To parameterize the coarse and fine level policy networks, we use ResNet [13] with 32 layers pretrained on the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) dataset [38]. We train the policy networks using 4 NVIDIA 1080ti GPUs.

Object Detectors Our coarse and fine level detectors use the YOLOv3 architecture [34], chosen for its reasonable trade off between accuracy on small objects and run-time performance. The backbone network, DarkNet-53, is pretrained on ImageNet. We train the detectors using a single NVIDIA 1080ti GPU.

5.3. Performance Metrics

We evaluate the performance using the following metrics: *average precision (AP)*, *average recall (AR)*, *average run-time per image (ms)*, and *ratio of sampled HR image*. For AP and AR, we compute the individual values across different categories for IoU=$\{.50, .55, .60, \ldots, .95\}$.

5.4. Experiments on xView Dataset

We evaluate the proposed approach and baseline methods on the xView dataset [22], which consists of large HR satellite images representing 60 categories. The training, validation, and test splits of the dataset have 846, 221, and 221 large scale images, respectively, with 3,000-6,000px in each dimension. The validation and test splits of the dataset have not been released publicly since the xView dataset was collected as part of the Object Detection challenge on Satellite Images. For this reason, we use the training split of the dataset to train the object detectors and policy networks and test the proposed framework. In particular, 47% and 12% of the large images are used to train and test the coarse and fine level detector whereas the remaining 41% is used to train the policy network. The policy network is then tested on the same 12% of the large images used to test the detectors. Among the 60 classes in the xView dataset, we test our efficient object detection model on the *small car* and *building* classes as they are the two classes represented by the most number of samples.

Coarse and Fine Level Detectors In the first step, we train the coarse and fine level detectors. We train the coarse and fine level detectors with a batch size of 8 for 65 and 87 epochs, respectively. The coarse and fine level detectors achieve 26.3% and 39.8% average precision (AP) and 39.0% and 60.9% average recall (AR). The coarse detector operates on the 64×64px LR images, x_l^j, whereas the fine level detector operates on the 320×320px HR images, x_h^j. In other words, 320×320px images, x_h^j, correspond to a subpatch of 600×600px images, x_H^i, where we set the overlap between subpatches to 40 pixels. The coarse and fine detectors run on average at 10 and 50 ms per image on a NVIDIA GeForce GTX 1080 Ti GPU.

Policy Networks In the second step, we train the policy networks and treat the detectors as *black boxes* as we keep their weights fixed. We form the initial large images by taking 2,400×2,400px crops out of the xView images, then divide them into 16 non-overlapping 600×600px patches resulting in 16 output units in CPNet. For CPNet, we downsample the images by a factor of about 5 to 448×448px. For FPNet, we use 4 output units to represent the subpatches in the patch selected by the CPNet. The subpatches have size 320×320px and an overlap of 40px. As the input, we use the 112×112px patches of the 448×448px images used by CPNet. Then, we train CPNet with a batch size of 512 for 643 epochs. In the final step, we train FPNet with a batch size of 512 for 459 epochs. For both networks, we set the learning rate to 1e-4 and the hyperparameters α, λ, σ and β to 0.8, 0.25, 0.25 and 0.05. All the networks are trained with the Adam optimizer [18].

Quantitative and Qualitative Analysis As shown in Table 1, the cascaded approach using CPNet+FPNet provides optimal results considering the run-time efficiency and image acquisition cost. It yields AP and AR scores only 0.9% and 1.2% lower, respectively, than the sliding window approach using HR images with the fine detector. Meanwhile, our approach uses HR images for 31.5% of the full area of interest on average and delivers $2.2\times$ higher run-time efficiency. On the other hand, the CPNet+FPNet approach in test time outperforms CPNet and FPNet only approaches

	Coarse Level				Fine Level				Coarse + Fine Level			
Model/Metric	AP	AR	Run-time	HR	AP	AR	Run-time	HR	AP	AR	Run-time	HR
Random (5×)	29.2	47.0	1770	43.7	27.2	49.3	1920	50	24.1	47.1	1408	31
Entropy (5×)	30.1	47.9	1766	43.7	28.3	50.1	1932	50	25.4	47.2	1415	31
Sliding Window-L (5×)	26.3	39.8	640	0	26.3	39.8	640	0	26.3	39.8	640	0
Sliding Window-H	39.0	60.9	3200	100	39.0	60.9	3200	100	39.0	60.9	3200	100
Gao et al. [7] (5×)	35.3	55.2	1780	40.5	35.2	55.8	1721	35.4	35.2	55.5	1551	31.6
Ours (5×)	38.2	59.8	1725	40.6	38.3	59.6	1683	35.5	38.1	59.7	1484	31.5

Table 1. Results for the *building* and *small car* classes. The coarse and fine level only methods refer to using only coarse and fine level policy network in test time. The coarse + fine level method first runs the coarse level policy network on initial large image, and fine level policy network is run on the images activated by the coarse network.

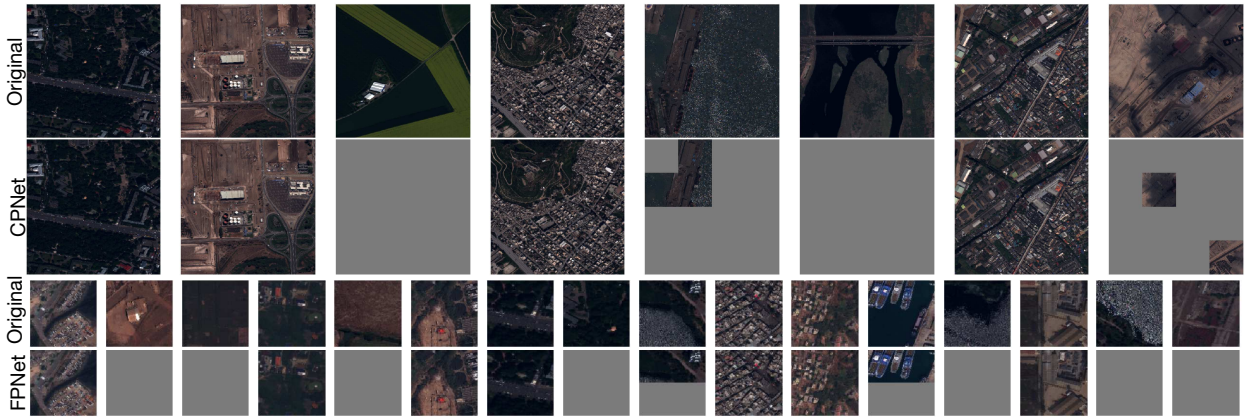

Figure 3. Visualization of the learned policies by the coarse and fine level policy networks. The top row shows the original large LR images given to the coarse policy network. The second row shows the patches chosen by the coarse policy network, and last two rows represent the policy learned by the fine level policy network. The coarse and fine level policy networks learn to sample LR-HR patches/sub-patches, respectively. The regions for using LR images are shown in grey.

in terms of the use of HR images and run-time efficiency. Additionally, Fig. 3 demonstrates the policies learned by CPNet and FPNet. For instance, in columns 1, 2 and 4, CPNet learns to use HR images as these images mostly contain small buildings and cars. On the other hand, it learns to use LR images when the image is populated with no objects or large buildings (columns 3 and 8). Similarly, FPNet chooses HR images when the subpatches are populated with small buildings or cars as in columns 1, 4, 10 and 11.

Finally, in Fig. 4 we show the probability of zoom-in for the CPNet and FPNet when the number and size of objects in each patch/subpatch increase. The probability of zoom-in increases w.r.t. number of objects because we scale R_{acc} with the number of objects N_i in a patch/subpatch. On the other hand, the increasing average size of the objects does not necessarily mean an increased probability of zoom-in. This is expected as the recall difference between the coarse and fine detector reduces with increasing size of the objects.

Ablation Experiments on xView Previously, we already performed ablation experiments (Table 1) by excluding coarse (CPNet) or fine level search (FPNet) from our cascaded approach. In this section, we want to quantify the effect of the coarse detector on CPNet, FPNet and the baseline methods. By removing the coarse level detector from Eq. 18, we can understand if the policy networks can learn the difference between *zooming-in when there is object of interest* and *zooming-in when the fine level detector outperforms coarse level detector*. Thus, we train the CPNet and FPNet using the modified reward function. In particular, we encourage the network to zoom-in when the fine detector achieves a positive recall value on a HR image.

Table 2 shows that removing the coarse detector improves the run-time efficiency slightly with the cost of using about 13% more HR images. This can be explained by the

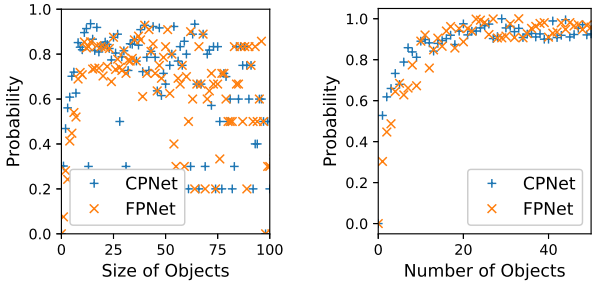

Figure 4. Visualization of when the CPNet and FPNet zooms-in w.r.t. the average number of objects and size of objects. Size corresponds to the normalized area of the objects in pixels.

	Coarse Level				Fine Level				Coarse + Fine Level			
Model/Metric	AP	AR	Run-time	HR	AP	AR	Run-time	HR	AP	AR	Run-time	HR
Random (5×)	22.7	32.4	1792	56.1	21.5	31.4	1601	50.0	19.9	29.8	1504	47.1
Entropy (5×)	23.0	32.2	1801	56.2	22.2	30.6	1605	50.0	20.5	31.1	1511	47.5
Sliding Window-H	39.0	60.9	3200	100	39.0	60.9	3200	100	39.0	60.9	3200	100
Ours (5×)	37.4	58.1	1882	58	37.5	58.2	1640	45.1	37.3	58.1	1421	44.1

Table 2. Results for the *building* and *small car* classes when the coarse detector is *disabled*. The coarse detector is removed during training and test time. We removed the method by Gao et al. [7] as its agent uses coarse detector to zoom-in.

fact that maintaining the accuracy requires zooming-in to more patches with the removal of coarse detector. In conclusion, by using the coarse detector we achieve better accuracy and similar efficiency while using fewer HR images.

5.5. Experiments on Caltech Pedestrian Dataset

Finally, we run experiments on the Caltech Pedestrian dataset (CPD) [5] to quantify the validity of the proposed approach on *traditional images*. We resize the original 640×480px images to 860×860px similarly to [7]. Following *reasonable* setting in CPD, we use two sets of 5,000 images for training the detectors and policy networks, and two sets of 3,000 images for validation and test experiments.

Coarse and Fine Level Detectors We first train the coarse and fine detectors. The fine detector is trained on 320×320px patches of the 860×860px images and the coarse detector is trained on downsampled versions of the patches.

Policy Network Since the images are $\sim 3\times$ smaller than xView images, we only use CPNet and form its action space with *9 unique patches*. Each patch has size 320×320px with 50px overlap between the patches. We use 172×172px images downsampled from 860×860px to maintain the same $5\times$ downsampling ratio as in our xView experiments. We set the other parameters similarly to our xView experiments and train the CPNet for 1450 epochs.

Quantitative Analysis As shown in Table 3, the proposed approach outperforms the baselines and state-of-the-art [7] by a large margin when using a $5\times$ downsampling ratio. When using $2\times$ downsampling, our method performs similarly to [7] in terms of AP and AR, but delivers higher run-time performance. This is because their agent uses coarse detection results over the entire image, whereas CPNet only chooses a binary action for each image patch without using coarse detection, thus requiring less overhead. Our approach can be even further optimized by running the patches through the coarse or fine detectors *in parallel*, whereas [7] performs *sequential* inference.

Finally, we measure the performance of CPNet w.r.t. different downsampling factors for the coarse detector. As seen in Fig. 5, the AP, AR and Run-time do not change drastically with decreasing the downsampling ratio whereas the number of zoom-ins decreases sharply, since the coarse detector performs better with finer images but also has higher run-time cost. These results show that CPNet can maintain high AP, AR and run-time efficiency when used with the coarse detector trained on images across different resolutions.

Model/Metric	AP	AR	Run-time	HR
Random (×5)	30.9	62.1	248	44.4
Entropy (×5)	34.0	63.9	250	44.4
Sliding Window-L (×5)	21.2	46.3	90	0
Sliding Window-H	64.7	74.7	450	100
Gao et al. [7] (×2)	64.5	73.1	295	7.1
Gao et al. [7] (×5)	57.3	70.7	309	43.3
CPNet (×2)	64.4	74.5	267	6.6
CPNet (×5)	61.7	74.1	270	44.4

Table 3. Results on the Caltech Pedestrian Dataset.

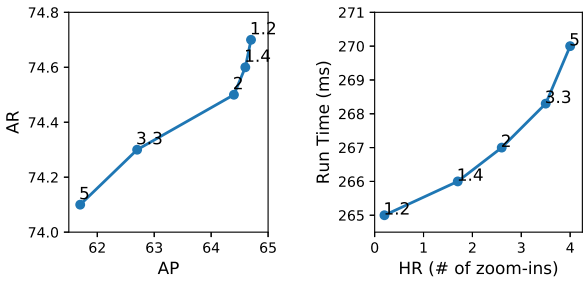

Figure 5. Performance of the CPNet when changing the resolution of the coarse detector on CPD. We use 172×172px images for the CPNet and only change the resolution of coarse detector.

6. Conclusion

In this study, we proposed an approach to efficiently process large images for object detection without changing the underlying structure of a detector. In particular, we trained two policy networks, CPNet and FPNet, using reinforcement learning with the dual reward of maintaining the accuracy while maximizing the use of LR images with a coarse detector. By choosing actions for the full image in one step, the policy networks introduce minimal overhead. Our experiments on the xView, consisting of very large satellite images, indicate that the proposed approach increases run-time efficiency by $2.2\times$ while reducing dependency on HR images by about 70%. Finally, our approach delivers 40% run-time increase on Caltech Pedestrian Dataset images.

References

[1] B. Alexe, N. Heess, Y. W. Teh, and V. Ferrari. Searching for objects driven by context. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 881–889, Stateline, NV, USA, Dec. 2012. Curran Associates, Inc.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[3] 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.

[4] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan. Vehicle detection in satellite images by hybrid deep convolutional neural networks. *IEEE Geoscience and remote sensing letters*, 11(10):1797–1801, 2014.

[5] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. 2009.

[6] M. Drusch, U. D. Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, A. Meygret, F. Spoto, O. Sy, F. Marchese, and P. Bargellini. Sentinel-2: Esa's optical high-resolution mission for gmes operational services. *Remote Sensing of Environment*, 120:25 – 36, 2012.

[7] M. Gao, R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Dynamic Zoom-in Network for Fast Object Detection in Large Images. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6926–6935, Salt Lake City, UT, USA, June 2018. IEEE.

[8] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, Columbus, OH, USA, June 2014.

[10] A. Gonzalez-Garcia, A. Vezhnevets, and V. Ferrari. An active search strategy for efficient object class detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3022–3031, Boston, MA, USA, June 2015. IEEE.

[11] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *2009 IEEE 12th International Conference on Computer Vision*, pages 237–244, Kyoto, Japan, Sept. 2009. IEEE.

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

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE.

[14] D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. *arXiv preprint arXiv:1901.09960*, 2019.

[15] M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[16] Z. Jie, X. Liang, J. Feng, X. Jin, W. Lu, and S. Yan. Tree-Structured Reinforcement Learning for Sequential Object Localization. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 127–135, Barcelona, Spain, Dec. 2016. Curran Associates, Inc.

[17] 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.

[18] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference for Learning Representations*, San Diego, CA, USA, May 2015.

[19] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Hager, A. Lukezic, A. Eldesokey, et al. The visual object tracking vot2017 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1949–1972, 2017.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] R. LaLonde, D. Zhang, and M. Shah. ClusterNet: Detecting Small Objects in Large Scenes by Exploiting Spatio-Temporal Information. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4003–4012, Salt Lake City, UT, USA, June 2018. IEEE.

[22] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord. xView: Objects in Context in Overhead Imagery. *arXiv preprint arXiv:1802.07856*, Feb. 2018.

[23] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[24] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5325–5334, Boston, MA, USA, June 2015. IEEE.

[25] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[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 B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Amsterdam, Netherlands, Oct. 2016. Springer International Publishing.

[28] Y. Lu, T. Javidi, and S. Lazebnik. Adaptive object detection using adjacency and zoom prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2351–2359, 2016.

[29] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2894–2902, 2016.

[30] Z. Meng, X. Fan, X. Chen, M. Chen, and Y. Tong. Detecting small signs from large images. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 217–224. IEEE, 2017.

[31] M. Najibi, B. Singh, and L. S. Davis. Autofocus: Efficient multi-scale inference. *arXiv preprint arXiv:1812.01600*, 2018.

[32] A. Pirinen and C. Sminchisescu. Deep reinforcement learning of region proposal networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6945–6954, 2018.

[33] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, Honolulu, HI, USA, July 2017. IEEE.

[34] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. Technical report, Apr. 2018.

[35] 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.

[36] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017.

[37] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024, 2017.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec. 2015.

[39] N. Sergievskiy and A. Ponamarev. Reduced focal loss: 1st place solution to xview object detection in satellite imagery. *arXiv preprint arXiv:1903.01347*, 2019.

[40] E. Sheehan, C. Meng, M. Tan, B. Uzkent, N. Jean, M. Burke, D. Lobell, and S. Ermon. Predicting economic development using geolocated wikipedia articles. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2698–2706. ACM, 2019.

[41] E. Sheehan, B. Uzkent, C. Meng, Z. Tang, M. Burke, D. Lobell, and S. Ermon. Learning to interpret satellite images using wikipedia. *arXiv preprint arXiv:1809.10236*, 2018.

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[44] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013.

[45] 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.

[46] B. Uzkent, M. J. Hoffman, and A. Vodacek. Real-time vehicle tracking in aerial video using hyperspectral features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 36–44, 2016.

[47] B. Uzkent, M. J. Hoffman, A. Vodacek, and B. Chen. Feature matching with an adaptive optical sensor in a ground target tracking system. *IEEE Sensors Journal*, 15(1):510–519, 2014.

[48] B. Uzkent, A. Rangnekar, and M. Hoffman. Aerial vehicle tracking by adaptive fusion of hyperspectral likelihood maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 39–48, 2017.

[49] B. Uzkent and Y. Seo. Enkcf: Ensemble of kernelized correlation filters for high-speed object tracking. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141. IEEE, 2018.

[50] B. Uzkent, E. Sheehan, C. Meng, Z. Tang, M. Burke, D. Lobell, and S. Ermon. Learning to interpret satellite images in global scale using wikipedia. *arXiv preprint arXiv:1905.02506*, 2019.

[51] C. J. C. H. Watkins and P. Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992.

[52] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele. Sliding-windows for rapid object class localization: A parallel technique. In *Joint Pattern Recognition Symposium*, pages 71–81. Springer, 2008.

[53] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

[54] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu. Traffic-sign detection and classification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2110–2118, 2016.