AutoFocus: Efficient Multi-Scale Inference

Mahyar Najibi * Bharat Singh * Larry S. Davis
University of Maryland, College Park
{najibi,bharat,lsd}@cs.umd.edu

Abstract

This paper describes AutoFocus, an efficient multi-scale inference algorithm for deep-learning based object detectors. Instead of processing an entire image pyramid, AutoFocus adopts a coarse to fine approach and only processes regions which are likely to contain small objects at finer scales. This is achieved by predicting category agnostic segmentation maps for small objects at coarser scales, called FocusPixels. FocusPixels can be predicted with high recall, and in many cases, they only cover a small fraction of the entire image. To make efficient use of FocusPixels, an algorithm is proposed which generates compact rectangular FocusChips which enclose FocusPixels. The detector is only applied inside FocusChips, which reduces computation while processing finer scales. Different types of error can arise when detections from FocusChips of multiple scales are combined, hence techniques to correct them are proposed. AutoFocus obtains an mAP of 47.9% (68.3% at 50% overlap) on the COCO test-dev set while processing 6.4 images per second on a Titan X (Pascal) GPU. This is 2.5× faster than our multi-scale baseline detector and matches its mAP. The number of pixels processed in the pyramid can be reduced by 5× with a 1% drop in mAP. AutoFocus obtains more than 10% mAP gain compared to RetinaNet but runs at the same speed with the same ResNet-101 backbone.

1. Introduction

Human vision is foveal and active [1, 21]. The fovea, which observes the world at high-resolution, only corresponds to 5 degrees of the total visual field [32]. Our lower resolution peripheral vision has a field of view of 110 degrees [63]. To find objects, our eyes perform saccadic movements which rely on peripheral vision [31]. When moving between different fixation points, the region in between is simply ignored, a phenomenon known as saccadic masking [7, 27, 50]. Hence, finding objects is an active process and the search time depends on the complexity of the scene. For example, locating a face in a portrait photograph would take much less time than finding every face in a crowded market.

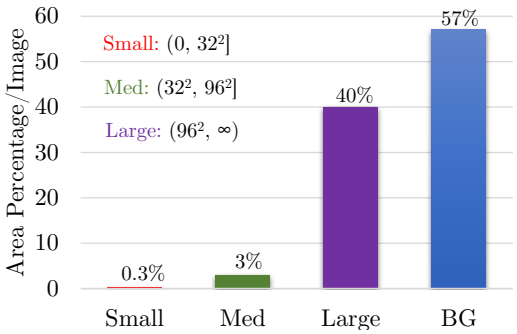


Figure 1: Area of objects of different sizes and the background in the COCO validation set. Objects are divided based on their area (in pixels) into small, medium, and large.

Adaptive processing, which is quite natural, brings several benefits. Many applications do not have real-time requirements and detectors are applied offline on billions of images/videos. Therefore, computational savings in a batch mode provide substantial monetary benefits. Examples include large-scale indexing of images and videos for visual search, APIs provided by cloud services, smart retail stores *etc*. While there is work on image classification which performs conditional computation [4, 39, 68], modern object detection algorithms perform static inference and process every pixel of a multi-scale image pyramid to detect objects of different sizes [60, 61, 54]. This is a very inefficient process as the algorithm spends equal energy at every pixel at different scales.

To provide some perspective, we show the percentage of pixels occupied per image for different size objects in the COCO dataset in Fig 1. Even though 40% of the object instances are small, they only occupy 0.3% of the area. If the image pyramid includes a scale of 3, then just to detect such a small fraction of the dataset, we end up performing 9 times more computation at finer-scales. If we add some padding around small objects to provide spatial context and only upsample these regions, their area would still

*Equal Contribution

be small compared to the resolution of the original image. So, when performing multi-scale inference, can we predict regions containing small objects from coarser scales?

If deep convolutional neural networks are an approximation of biological vision, it should be possible to localize object-like regions at lower resolution and recognize them by zooming on them at higher resolution - similar to the way our peripheral vision is coupled with foveal vision. To this end, we propose an object detection framework called *AutoFocus*, which adopts a coarse to fine approach and learns where to look in the next (larger) scale in the image pyramid. Thus, it saves computation while processing finer scales. This is achieved by predicting category agnostic binary segmentation maps for small objects, which we refer to as *FocusPixels*. A simple algorithm which operates on FocusPixels is designed to generate chips for the next image scale. AutoFocus only processes 20% of the image at the largest scale in the pyramid on the COCO dataset, without any drop in performance. This can be improved to as little as 5% with a 1% drop in performance.

2. Related Work

Image pyramids [67] and convolutional neural networks [34] are fundamental building blocks in the computer vision pipeline. Unfortunately, convolutional neural networks are not scale invariant. Therefore, for instance-level visual recognition problems, to recognize objects of different sizes, it is beneficial to rely on image pyramids [60]. While efficient training solutions have been proposed for multi-scale training [61], inference on image pyramids remains a computational bottleneck which prohibits their use in practice. Recently, a few methods have been proposed to accelerate multi-scale inference, but they have only been evaluated under constrained settings like pedestrian/face detection or object detection in videos [22, 62, 12, 59, 40, 30]. In this work, we propose a simple and pragmatic framework to accelerate multi-scale inference for generic object detection which is evaluated on benchmark datasets.

Accelerating object detection has a long history in computer vision. The Viola-Jones detector [65] is a classic example. It rejects easy regions with simple filters and spends more energy on promising object-like regions to accelerate the process. Several methods since then have been proposed to improve it [6, 73, 71]. Prior to deep-learning based object detectors, it was common to employ a multi-scale approach for object detection [66, 14, 18, 20, 17, 3] and several effective solutions were proposed to accelerate detection on image pyramids. Common techniques involved approximation of features to reduce the number of scales [17, 3], cascades [6, 16] or feature pyramids [15]. Recently, feature-pyramids have been extensively studied and employed in deep learning based object detectors as the representation provides a boost in accuracy without compromising speed [45, 43, 70, 9, 36, 44, 52, 24, 37, 41]. Although the use of image pyramids is common in challenge winning entries which primarily focus on performance [25, 13, 54, 41], efficient detectors which operate on a single low-resolution image (*e.g.* YOLO [56], SSD [43], RetinaNet [37]) are commonly deployed in practice. This is because multi-scale inference on pyramids of high-resolution images is prohibitively expensive.

AutoFocus alleviates this problem to a large extent and is designed to provide a smooth trade-off between speed and accuracy. It shows that it is possible to predict the presence of a small object at a coarser scale (referred to as FocusPixels) which enables avoiding computation in large regions of the image at finer scales. These are different from object proposals [11, 64, 58] where region candidates need to have a tight overlap with objects. Learning to predict FocusPixels is an easier task and does not require instance-level reasoning. AutoFocus shares the motivation with saliency and reinforcement learning based methods which perform a guided search while processing images [28, 26, 42, 53, 23, 49, 55, 29, 47], but it is designed to predict small objects in coarser scales and they need not be salient.

3. Background

We provide a brief overview of SNIP, which is the multi-scale training and inference method described in [60]. The core idea is to restrict the training samples to be in a pre-defined scale range which is appropriate for the input scale. For example, the detector is only trained on small objects at high resolution (larger scale) and large objects at low resolution (smaller scale). Because it is not trained on large objects at high resolution images, it is unlikely to detect them during inference as well. Rules are also defined to ignore large detections in high-resolution images during inference and vice-versa. Therefore, while merging detections from multiple scales, SNIP simply ignores large detections in high resolution images which contain most of the pixels.

Since the size of objects is known during training, it is possible to ignore large regions of the image pyramid by only processing appropriate context regions around objects. SNIPER [61] showed that training on such low resolution *chips* with appropriate scaling does not lead to any drop in performance when compared to training on full-resolution images. If we can automatically predict these chips for small objects at a coarser scale, we may not need to process the entire high-resolution image during inference as well. But when these chips are generated during training, many object instances get cropped and their size changes. This did not hurt performance during training and can also be regarded as a data augmentation strategy. Unfortunately, if chips are generated during inference and an object is cropped into multiple parts, it would increase the error rate.

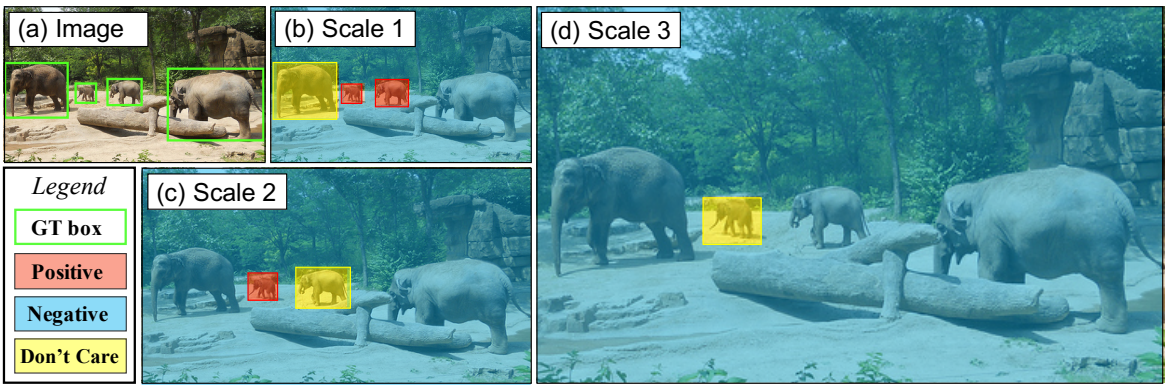


Figure 2: The figure illustrates how FocusPixels are assigned at multiple scales of an image. At scale 1 (b), the smallest two elephants generate FocusPixels, the largest one is marked as background and the one on the left is ignored during training to avoid penalizing the network for borderline cases (see Sec. 4.1 for assignment details). The labelling changes at scales 2 and 3 as the objects occupy more pixels. For example, only the smallest elephant would generate FocusPixels at scale 2 and the largest two elephants would generate negative labels.

So, apart from predicting where to look at the next scale, we also need to design an algorithm which correctly merges detections from chips at multiple scales.

4. The AutoFocus Framework

Classic features like SIFT [46] / SURF [2], combine two major components - the detector and the descriptor. The detector typically involved lightweight operators like Difference of Gaussians (DoG) [48], Harris Affine [51], Laplacian of Gaussians (LoG) [8] *etc*. The detector was applied on the entire image to find *interesting* regions. Therefore, the descriptor, which was computationally expensive, only needed to be computed for these interesting regions. This cascaded model of processing the image made the entire pipeline efficient.

Likewise, the AutoFocus framework is designed to predict interesting regions in the image and discards regions which are unlikely to contain objects at the next scale. It zooms and crops only such interesting regions when applying the detector at successive scales. AutoFocus is comprised of three main components: the first learns to predict *FocusPixels*, the second generates *FocusChips* for efficient inference and the third merges detections from multiple scales, which we refer to as *focus stacking* for object detection.

4.1. FocusPixels

FocusPixels are defined at the granularity of the convolutional feature map (like conv5). A pixel in the feature map is labelled as a FocusPixel if it has any overlap with a small object. An object is considered to be small if it falls in an area range (between 5×5 and 64×64 pixels in our implementation) in the resized chip (Sec. 4.2) which is input to the network . To train our network, we mark FocusPixels as positives. We also define some pixels in the feature map as invalid. Those pixels which overlap objects that have an area smaller or slightly larger than those defined as small are considered invalid (smaller than 5×5 or between 64×64 and 90×90). All other pixels are considered as negatives. AutoFocus is trained to generate high activations on regions which contain FocusPixels.

Formally, given an image of size $X \times Y$, and a fully convolutional neural network whose stride is s, then the labels L will be of size $X' \times Y'$, where $X' = \lceil \frac{X}{s} \rceil$ and $Y' = \lceil \frac{Y}{s} \rceil$. Since the stride is s, each label $l \in L$ corresponds to $s \times s$ pixels in the image. The label l is defined as follows,

$$l = \begin{cases} 1, & IoU(GT, l) > 0, a < \sqrt{GTArea} < b \\ -1, & IoU(GT, l) > 0, \sqrt{GTArea} < a \\ -1, & IoU(GT, l) > 0, b < \sqrt{GTArea} < c \\ 0, & \text{otherwise} \end{cases}$$

where IoU is intersection over union of the $s \times s$ label block with the ground truth bounding box. $GTArea$ is the area of the ground truth bounding box after scaling. a is typically 5, b is 64 and c is 90. If multiple ground-truth bounding boxes overlap with a pixel, FocusPixels ($l = 1$) are given precedence. Since our network is trained on 512×512 pixel chips, the ratio between positive and negative pixels is around 10, so we do not perform any re-weighting for the loss. Note that during multi-scale training, the same ground-truth could generate a label of 1, 0 or -1 depending on how much it has been scaled. The reason we regard pixels for medium objects as invalid ($l = -1$) is that the transition from small to large objects is not visually obvious. Extremely small objects in each scale are also marked

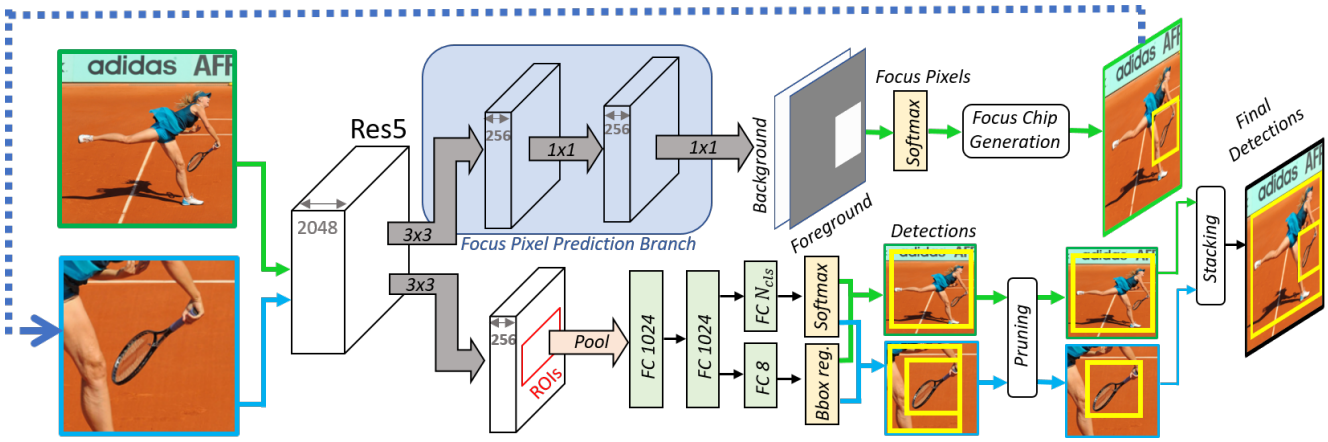


Figure 3: The figure illustrates how AutoFocus detects a person and a racket in an image. The green borders and arrows are for inference at the original resolution. The blue borders and arrows are shown when inference is performed inside FocusChips. In the first iteration, the network detects the person and also generates a heat-map to mark regions containing small objects. This is depicted in the white/grey map - it is used to generate FocusChips. In the next iteration, the detector is then applied inside FocusChips only. Inside FocusChips, there could be detections for the cropped object present at the larger resolution. Such detections are pruned and finally valid detections are stacked across multiple scales.

as invalid because after the early down-sampling operations, the network does not have sufficient information to make a correct prediction about them at that particular scale. The labelling scheme is visually depicted in Fig 2. For training the network, we add two convolutional layers (3×3 and 1×1) with a ReLU non-linearity on top of the conv5 feature-map. Finally, we have a binary softmax classifier to predict FocusPixels, shown in Fig 3.

4.2. FocusChip Generation

During inference, we mark those pixels $\mathcal{P}$ in the output as FocusPixels, where the probability of foreground is greater than a threshold t, which is a parameter controlling the speed-up and can be set with respect to the desired speed accuracy trade-off. This generates a number of connected components $\mathcal{S}$. We dilate each component with a filter of size $d \times d$ to increase contextual information needed for recognition. After dilation, components which become connected are merged. Then, we generate chips $\mathcal{C}$ which enclose these connected components. Note that chips of two connected components could overlap. As a result, these chips are merged and overlapping chips are replaced with their enclosing bounding-boxes. Some connected components could be very small, and may lack the contextual information needed to perform recognition. Many small chips also increase fragmentation which results in a wide range of chip sizes. This makes batch-inference inefficient. To avoid these problems, we ensure that the height and width of a chip is greater than a minimum size k. This process is described in Algorithm 1. Finally, we perform multi-scale inference on an image pyramid but successively prune regions which are unlikely to contain objects.

Algorithm 1: FocusChip Generator

Input : Predictions for feature map $\mathcal{P}$, threshold t, dilation constant d, minimum size of chip k
Output: Chips $\mathcal{C}$

1 Transform $\mathcal{P}$ into a binary map using the threshold t
2 Dilate $\mathcal{P}$ with a $d \times d$ filter
3 Obtain a set of connected components $\mathcal{S}$ from $\mathcal{P}$
4 Generate enclosing chips $\mathcal{C}$ of size $> k$ for each component in $\mathcal{S}$
5 Merge chips $\mathcal{C}$ if they overlap
6 **return** *Chips* $\mathcal{C}$

4.3. Focus Stacking for Object Detection

One issue with such cascaded multi-scale inference is that some detections at the boundary of the chips can be generated for cropped objects which were originally large. At the next scale, due to cropping, they could become small and generate false positives, such as the detections for the horse and the horse rider on the right, shown in Fig 4 (c). To alleviate this effect, Step 2 in Algorithm 1 is very important. Note that when we dilate the map $\mathcal{P}$ and generate chips, this ensures that no *interesting* object at the next scale would be observed at the boundaries of the chip (unless the chip shares a border with the image boundary). Otherwise, it would be enclosed by the chip, as these are generated around the dilated maps. Therefore, if a detection in the zoomed-in chip is observed at the boundary, we discard it,

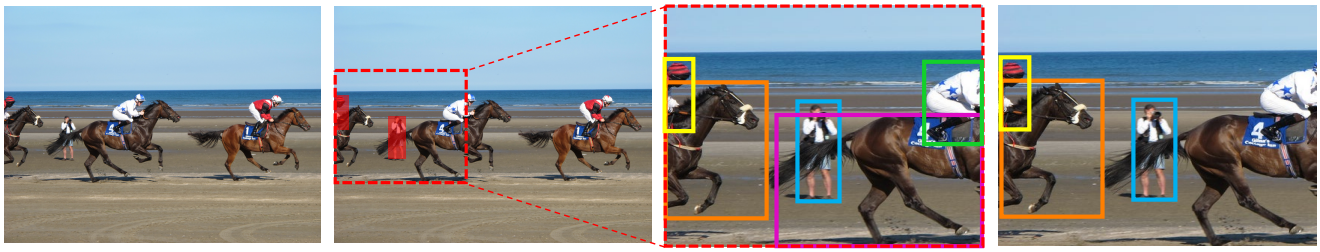


Figure 4: Pruning detections while FocusStacking. (a) Original Image (b) The predicted FocusPixels and the generated FocusChip (c) Detection output by the network (d) Final detections for the FocusChip after pruning.

even if it is within valid SNIP ranges, such as the horse rider eliminated in Fig 4 (d).

There are some corner cases when the detection is at the boundary (or boundaries x, y) of the image. If the chip shares one boundary with the image, we still check if the other side of the detection is completely enclosed inside or not. If it is not, we discard it, else we keep it. In another case, if the chip shares both the sides with the image boundary and so does the detection, then we keep the detection.

Once valid detections from each scale are obtained using the above rules, we merge detections from all the scales by projecting them to the image co-ordinates after applying appropriate scaling and translation. Finally, Non-Maximum Suppression is applied to aggregate the detections. The network architecture and an example of multi-scale inference and focus stacking is shown in Fig 3.

5. Datasets and Experiments

We evaluate AutoFocus on the COCO [38] and the PASCAL VOC [19] datasets. As our baseline, we use the SNIPER detector[1] [61] which obtains an mAP of 47.9% (68.3% at 50% overlap) on the COCO test-dev set and 47.5% (67.9% at 50% overlap) on the COCO validation set. We add the fully convolutional layers for AutoFocus which predict the FocusPixels. No other changes are made to the architecture or the training schedule. We use Soft-NMS [5] at test-time for Focus Stacking with $\sigma = 0.55$. Following SNIPER [61], the resolutions used for the 3 scales at inference are S1=$(480, 512)$, S2=$(800, 1280)$, and S3=$(1400, 2000)$. The first resolution is the minimum size of a side and the second one is the maximum in pixels. The scales corresponding to these resolutions are referred to as scales 1, 2 and 3 respectively in the following sections.

Since FocusChips of different size are generated, we group chips which are of similar size and aspect ratio to achieve a high batch inference throughput. In some cases, we need to perform padding when performing batch inference, which can slightly change the number of pixels processed per image. For large datasets, this overhead is negligible as the number of groups (for size and aspect ratio) can be increased without reducing the batch size.

[1] http://www.github.com/mahyarnajibi/SNIPER

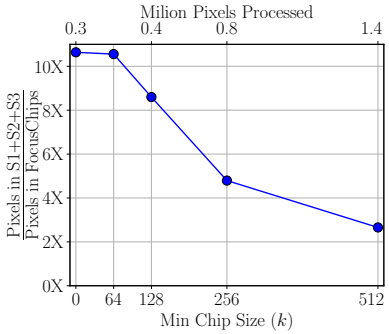
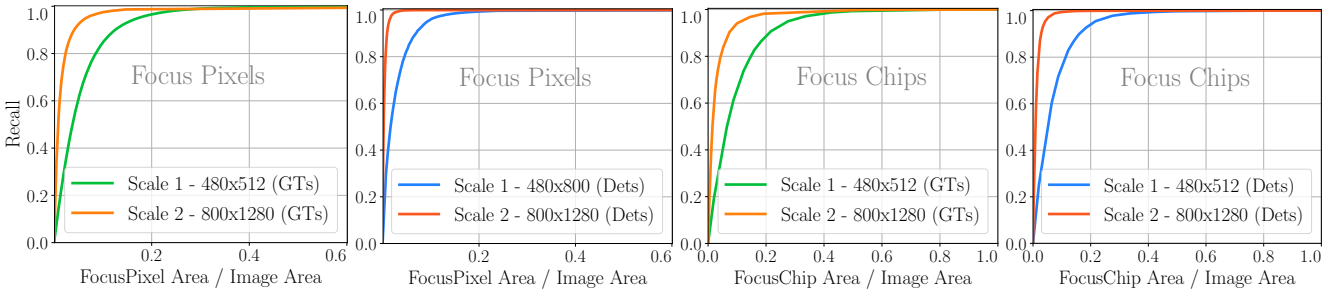


Figure 5: Upper-bound on the speed-up using FocusChips generated from optimal FocusPixels.

5.1. Stats for FocusPixels and FocusChips

In high resolution images (scale 3), the percentage of FocusPixels is very low (*i.e.* $\sim 4\%$). So, ideally a very small part of the image needs to be processed at high resolution. Since the image is upsampled, the FocusPixels projected on the image occupy an area of 63^2 pixels on average (the highest resolution images have an area of 1602^2 pixels on average). At lower scales (like scale 2), although the percentage of FocusPixels increases to $\sim 11\%$, their projections only occupy an area of 102^2 pixels on average (each image at this scale has an average area of 940^2 pixels). After dilating FocusPixels with a kernel of size 3×3, their percentages at scale 3 and scale 2 change to 7% and 18% respectively.

Using the chip generation algorithm, for a given minimum chip size (like $k = 512$), we also compute the upper bound on the speedup which can be obtained. This is under the assumption that FocusPixels can be predicted without any error (*i.e.* based on GTs). The bound for the speedup can change as we change the minimum chip size in the al-



(a) (b) (c) (d)

Figure 6: Quality of the FocusPixels and FocusChips. The x-axis represents the ratio of the area of FocusPixels or FocusChips to that of the image. The y-axis changes as follows, (a) FocusPixel recall is computed based on the GT boxes (b) FocusPixel recall is computed using the confident detections (c) FocusChip recall is computed based on the GT boxes (d) FocusChip recall is computed based on the confident detections.

gorithm. Fig 5 shows the effect of the minimum chip size parameter k for FocusChip generation in algorithm 1. The same value is used at each scale. For example, reducing the minimum chip size from 512 to 64 can lead to a theoretical speedup of $\sim$ 10 times over the baseline which performs inference on 3 scales. However, a significant reduction in minimum chip size can also affect detection performance - a reasonable amount of context is necessary for retaining high detection accuracy.

5.2. Quality of FocusPixel prediction

We evaluate how well our network predicts FocusPixels at different scales. To measure the performance, we use two criteria. First, we measure recall for predicting FocusPixels at two different resolutions. This is shown in Fig 6 a. This gives us an upper bound on how accurately we localize small objects using low resolution images. However, not all ground-truth objects which are annotated might be correctly detected. Note that our eventual goal is to accelerate the detector. Therefore, if we crop a region in the image which contains a ground-truth instance but the detector is not able to detect it, cropping that region would not be useful. The final effectiveness of FocusChips is coupled with the detector, hence we also evaluate the accuracy of FocusPixel prediction on regions which are confidently detected as shown in Fig 6 b. To this end, we only consider FocusPixels corresponding to those GT boxes which are covered (IoU $>$ 0.5) by a detection with a score greater than 0.5. At a threshold of 0.5, the detector still obtains an mAP of 47% which is within 1% of the final mAP and does not have a high false positive rate.

As expected, we obtain better recall at higher resolutions with both metrics. We can cover all confident detections at the higher resolution (scale 2) when the predicted FocusPixels cover just 5% of total image area. At a lower resolution (scale 1), when the FocusPixels cover 25% of the total image area, we cover all confident detections, see Fig 6 b.

5.3. Quality of FocusChips

While FocusPixels are sufficient to generate enclosing regions which need to be processed, current software implementations require the input image to be a rectangle for efficient processing. To this end, we evaluate the performance of the enclosing chips generated using the FocusPixels. Similar to Section 5.2, we use two metrics - one is recall of all GT boxes which are enclosed by FocusChips, the other one is recall for GT boxes enclosed by FocusChips which have a confident overlapping detection. To achieve perfect recall for confident detections at scale 2, FocusChips cover 5% more area than FocusPixels. At scale 1, they cover 10% more area. This is because objects are often not rectangular in shape. These results are shown in Fig 6 d.

5.4. Speed Accuracy Trade-off

We perform grid-search on different parameters, which are dilation, min-chip size and the threshold to generate FocusChips on a subset of 100 images in the validation set. For a given average number of pixels, we check which configuration of parameters obtains the best mAP on this subset. Since there are two scales at which we predict FocusPixels, we first find the parameters of AutoFocus when it is only applied to the highest resolution scale. Then we fix these parameters for the highest scale, and find parameters for applying AutoFocus at scale 2.

In Fig 7 we show that the multi-scale inference baseline which uses 3 scales obtains an mAP of 47.5% (and 68% at 50% overlap) on the val-2017 set. Using only the lower two scales obtains an mAP of 45.4%. The middle scale alone obtains an mAP of 37%. This is partly because the detector is trained with the scale normalization scheme proposed in [60]. As a result, the performance on a single scale alone is not very good, although multi-scale performance is high.

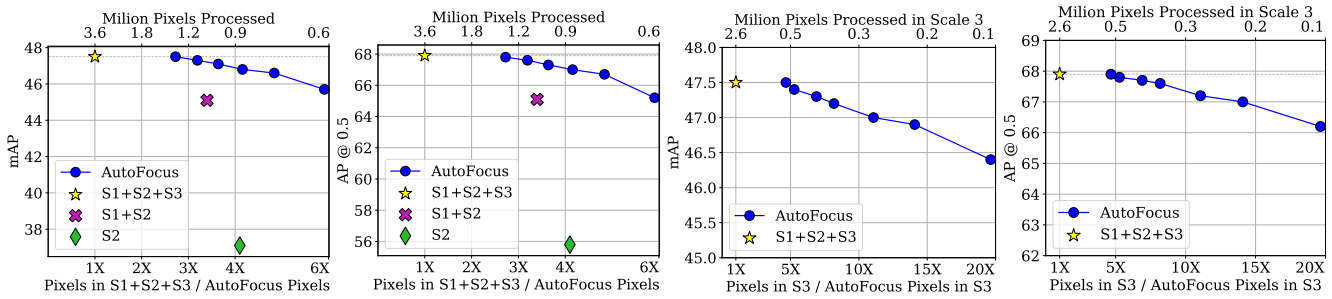


Figure 7: Results are on the val-2017 set. (a,c) show the mAP averaged for IoU from 0.5 to 0.95 with an interval of 0.05 (COCO metric). (b,d) show mAP at 50% overlap (PASCAL metric). We can reduce the number of pixels processed by a factor of 2.8 times without any loss of performance. A 5 times reduction in pixels is obtained with a drop of 1% in mAP.

The maximum savings in pixels which we can obtain while retaining performance is 2.8 times. We lose approximately 1% mAP to obtain a 5 times reduction over our baseline in the val-2017 set.

We also perform an ablation experiment for the Focus-Pixels predicted using scale 2. Note that the performance of just using scales 1 and 2 is 45%. We can retain the original performance of 47.5% on the val-2017 set by processing just one fifth of scale 3. With a 0.5% drop we can reduce the pixels processed by 11 times in the highest resolution image. This can be improved to 20 times with a 1% drop in mAP, which is still 1.5% better than the performance of the lower two scales.

Method	Pixels	AP	AP^{50}	S	M	L
Retina [37]	950^2	37.8	57.5	20.2	41.1	49.2
LightH [35]	940^2	41.5	-	25.2	45.3	53.1
Refine+ [72]	3100^2	41.8	62.9	25.6	45.1	54.1
Corner+ [33]	1240^2	42.1	57.8	20.8	44.8	56.7
SNIPER [61]	1910^2	**47.9**	**68.3**	**31.5**	**50.5**	**60.3**
AutoFocus	1175^2	**47.9**	**68.3**	**31.5**	**50.5**	**60.3**
	930^2	47.2	67.5	30.9	49.0	60.0
	860^2	46.9	67.0	30.1	48.9	60.0

Table 1: Comparison with SNIPER on the COCO test-dev. This is our multi-scale baseline. Results for others are taken from the papers/GitHub of the authors. Note that average pixels processed over the dataset are reported (instead of the shorter side). All methods use a ResNet-101 backbone. '+' denotes the multi-scale version provided by the authors.

Results on the COCO test-dev set are provided in Table 1. **While matching SNIPER's performance of 47.9% (68.3% at 0.5 IoU), AutoFocus processes 6.4 images per second on the test-dev set with a Titan X Pascal GPU.** SNIPER processes 2.5 images per second. RetinaNet with a ResNet-101 backbone and a FPN architecture processes

Method	Pixels	AP^{50}	AP^{70}
Deformable ConvNet [13]	705^2	82.3	67.8
Deformable ConvNet v2 [74]	705^2	84.9	73.5
SNIPER [61]	1915^2	86.6	80.5
AutoFocus*	860^2	85.8	79.5
AutoFocus	700^2	85.3	78.1
	1250^2	86.5	80.2

Table 2: Comparison on PASCAL VOC 2007 test-set. All methods use ResNet-101 and trained on VOC2012 trainval+VOC2007 trainval. The average pixels processed over the dataset are also reported. To show the robustness of AutoFocus to hyper-parameter choices, in '*' we use the same parameters as COCO and run the algorithm on PASCAL.

6.3 images per second on a P100 GPU (which is like Titan X), but obtains 37.8% mAP [2]. We also report the number of pixels processed with a few efficient recent detectors. Detectors which perform better than SNIPER like MegDet [54] or PANet [41] are slower because they use complex architectures like ResNext-152 [69] etc. To the best of our knowledge, AutoFocus is the fastest detector which obtains an mAP of 47.9% (or 68.3% at 0.5 IoU) on the COCO dataset. We show the inference process for AutoFocus on a few images in the COCO val-2017 set in Fig 8. We also report results on the PASCAL VOC dataset in Table 2. To show the robustness of AutoFocus to its hyper-parameters, we use exactly the same hyper-parameters tuned for COCO (shown as AutoFocus*). While processing the same area as DeformableV2 [74], AutoFocus achieves 4.6% better AP at 0.7 IoU. It also matches the performance of SNIPER while being considerably more efficient. Its mAP (on the COCO metric) can be further improved by using refinement techniques like cascade-RCNN [10].

[2] https://github.com/facebookresearch/Detectron/blob/master/MODEL_ZOO.md

Figure 8: Each column shows the inference pipeline in AutoFocus. The confidence for FocusPixels and FocusChips are shown in red, and yellow respectively in the second and fourth rows. Detections are shown in green. As can be seen, complex images containing many small objects like the two leftmost columns can generate multiple FocusChips in high resolutions like 1400×2000. Images which do not contain small objects are not processed at all in high resolution, like the one in the rightmost column.

6. Future Work

While results for only a Faster R-CNN based detector were presented, it is possible to combine AutoFocus with detectors like YOLOv2 [57], RetinaNet [37] and for other instance-level recognition tasks like pose estimation, instance segmentation *etc*. It would also be of interest to develop efficient multi-scale algorithms for tasks like stuff segmentation. Combining tracking and multi-scale inference can lead to further acceleration in videos.

Acknowledgement The research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via DOI/IBC Contract Numbers D17PC00287 and D17PC00345. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of IARPA, DOI/IBC or the U.S. Government. The authors would also like to thank an Amazon Machine Learning gift for the AWS credits used for this research.

References

[1] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International journal of computer vision*, 1(4):333–356, 1988. 1

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006. 3

[3] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2903–2910. IEEE, 2012. 2

[4] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup. Conditional computation in neural networks for faster models. *ICML Workshop on Abstraction in Reinforcement Learning*, 2016. 1

[5] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-nms – improving object detection with one line of code. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5562–5570. IEEE, 2017. 5

[6] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243. IEEE, 2005. 2

[7] B. G. Breitmeyer and L. Ganz. Implications of sustained and transient channels for theories of visual pattern masking, saccadic suppression, and information processing. *Psychological review*, 83(1):1, 1976. 1

[8] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. In *Readings in Computer Vision*, pages 671–679. Elsevier, 1987. 3

[9] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer, 2016. 2

[10] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. *CVPR*, 2018. 7

[11] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3241–3248. IEEE, 2010. 2

[12] K. Chen, J. Wang, S. Yang, X. Zhang, Y. Xiong, C. Change Loy, and D. Lin. Optimizing video object detection via a scale-time lattice. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7814–7823, 2018. 2

[13] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *ICCV*, 2017. 2, 7

[14] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 2

[15] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014. 2

[16] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *Computer Vision–ECCV 2012*, pages 645–659. Springer, 2012. 2

[17] P. Dollár, S. J. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *BMVC*, volume 2, page 7. Citeseer, 2010. 2

[18] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. *BMVC*, 2009. 2

[19] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 5

[20] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 2

[21] J. M. Findlay and I. D. Gilchrist. *Active vision: The psychology of looking and seeing*. Number 37. Oxford University Press, 2003. 1

[22] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[23] A. Gonzalez-Garcia, A. Vezhnevets, and V. Ferrari. An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3022–3031, 2015. 2

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 2

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE con-*

ference on computer vision and pattern recognition, pages 770–778, 2016. 2

[26] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 2

[27] D. E. Irwin, J. S. Brown, and J.-s. Sun. Visual masking and visual integration across saccadic eye movements. *Journal of Experimental Psychology: General*, 117(3):276, 1988. 1

[28] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998. 2

[29] 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. 2

[30] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, 2017. 2

[31] W. Kienzle, M. O. Franz, B. Schölkopf, and F. A. Wichmann. Center-surround patterns emerge as optimal predictors for human saccade targets. *Journal of vision*, 9(5):7–7, 2009. 1

[32] H. Kolb. Simple anatomy of the retina. 1995. 1

[33] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 7

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[35] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017. 7

[36] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. 2

[37] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2, 7, 9

[38] 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. 5

[39] L. Liu and J. Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. *AAAI*, 2018. 1

[40] M. Liu and M. Zhu. Mobile video object detection with temporally-aware feature maps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[41] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018. 2, 7

[42] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum. Learning to detect a salient object. *IEEE Transactions on Pattern analysis and machine intelligence*, 33(2):353–367, 2011. 2

[43] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[44] Y. Liu, H. Li, J. Yan, F. Wei, X. Wang, and X. Tang. Recurrent scale approximation for object detection in cnn. In *IEEE international conference on computer vision*, volume 5, 2017. 2

[45] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2

[46] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 3

[47] 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. 2

[48] D. Marr and E. Hildreth. Theory of edge detection. *Proc. R. Soc. Lond. B*, 207(1167):187–217, 1980. 3

[49] 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. 2

[50] E. Matin. Saccadic suppression: a review and an analysis. *Psychological bulletin*, 81(12):899, 1974. 1

[51] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004. 3

[52] M. Najibi, P. Samangouei, R. Chellappa, and L. Davis. SSH: Single stage headless face detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4875–4884, 2017. 2

[53] M. Najibi, F. Yang, Q. Wang, and R. Piramuthu. Towards the success rate of one: Real-time unconstrained salient object detection. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1432–1441. IEEE, 2018. 2

[54] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. Megdet: A large mini-batch object detector. *CVPR*, 2018. 1, 2, 7

[55] 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. 2

[56] 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. 2

[57] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE, 2017. 9

[58] 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. 2

[59] M. J. Shafiee, B. Chywl, F. Li, and A. Wong. Fast yolo: A fast you only look once system for real-time embedded object detection in video. *arXiv preprint arXiv:1709.05943*, 2017. 2

[60] B. Singh and L. S. Davis. An analysis of scale invariance in object detection-snip. *CVPR*, 2018. 1, 2, 6

[61] B. Singh, M. Najibi, and L. S. Davis. SNIPER: Efficient multi-scale training. *NIPS*, 2018. 1, 2, 5, 7

[62] G. Song, Y. Liu, M. Jiang, Y. Wang, J. Yan, and B. Leng. Beyond trade-off: Accelerate fcn-based face detector with higher accuracy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7756–7764, 2018. 2

[63] H. Strasburger, I. Rentschler, and M. Jüttner. Peripheral vision and pattern recognition: A review. *Journal of vision*, 11(5):13–13, 2011. 1

[64] 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. 2

[65] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001. 2

[66] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005. 2

[67] A. Witkin. Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 150–153. IEEE, 1984. 2

[68] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris. Blockdrop: Dynamic inference paths in residual networks. *CVPR*, 2018. 1

[69] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017. 7

[70] F. Yang, W. Choi, and Y. Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2016. 2

[71] C. Zhang and P. A. Viola. Multiple-instance pruning for learning efficient cascade detectors. In *Advances in neural information processing systems*, pages 1681–1688, 2008. 2

[72] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Single-shot refinement neural network for object detection. *CVPR*, 2018. 7

[73] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006. 2

[74] X. Zhu, H. Hu, S. Lin, and J. Dai. Deformable convnets v2: More deformable, better results. *arXiv preprint arXiv:1811.11168*, 2018. 7