Priming Neural Networks

Amir Rosenfeld , Mahdi Biparva , and John K.Tsotsos

Department of Electrical Engineering and Computer Science
York University
Toronto, ON, Canada, M3J 1P3
{amir, mhdbprv, tsotsos}@cse.yorku.ca

Abstract

Visual priming is known to affect the human visual system to allow detection of scene elements, even those that may have been near unnoticeable before, such as the presence of camouflaged animals. This process has been shown to be an effect of top-down signaling in the visual system triggered by the said cue. In this paper, we propose a mechanism to mimic the process of priming in the context of object detection and segmentation. We view priming as having a modulatory, cue dependent effect on layers of features within a network. Our results show how such a process can be complementary to, and at times more effective than simple post-processing applied to the output of the network, notably so in cases where the object is hard to detect such as in severe noise, small size or atypical appearance. Moreover, we find the effects of priming are sometimes stronger when early visual layers are affected. Overall, our experiments confirm that top-down signals can go a long way in improving object detection and segmentation.

Figure 1: Visual priming: something is hidden in plain sight in this image. It is unlikely to notice it without a cue on what it is (for an observer that has not seen this image before). Once a cue is given, perception is modified to allow successful detection. See footnote at bottom of this page for the cue, and supplementary material for the full answer.

1. Introduction

Psychophysical and neurophysiological studies of the human visual system confirm the abundance of top-down effects that occur when an image is observed. Such top-down signals can stem from either internal (endogenous) processes of reasoning and attention or external (exogenous) stimuli - i.e. cues - that affect perception (cf. [35], Chapter 3 for a more detailed breakdown). External stimuli having such effects are said to *prime* the visual system, and potentially have a profound effect on an observer's perception. This often results in an "Aha!" moment for the viewer, as he/she suddenly perceives the image differently; Fig. 1 shows an example of such a case. We make here the distinction between 3 detection strategies: (1) *free viewing*, (2) *priming* and (3) *pruning*. Freely viewing the image, the default strategy, likely reveals nothing more than a dry grassy field near a house. Introducing a cue about a target in the image[1] results in one of two possibilities. The first, also known as priming, is modification to the computation performed when viewing the scene with the cue in mind. The second, which we call pruning - is a modification to the decision process after all the computation is finished. When the task is to detect objects, this can mean retaining all detections match the cue, even very low confidence ones and discarding all others. While both are viable ways to incorporate the knowledge brought on by the cue,

[1] Object in image: ɹǝɔ

priming often highly increases the chance of detecting the cued object. Viewing the image for an unlimited amount of time and pruning the results is less effective; in some cases, detection is facilitated only by the cue. We claim that priming allows the cue to affect the visual process from early layers, allowing detection where it was previously unlikely to occur in free-viewing conditions. This has also recently gained some neurophysiological evidence [2].

In general, we view priming and pruning as complementary processes on the output of a system: let us represent the output of the system as

$$y = F(I; w) \tag{1}$$

where I is some input, F is whatever function the system performs (e.g., classification, detection, segmentation) and w are the parameters of F. Given some external hint h to the system allows to modify the output of y using an auxiliary function G. In pruning, we express this as:

$$y' = G(F(I; w), h) \tag{2}$$

On the other hand, priming can be expressed as:

$$y'' = F(I; G(w, h)) \tag{3}$$

In other words, while pruning (Eq. 2) operates on the output of F, priming (Eq. 3) changes the computation in F in a more fundamental way, dependent on h.

In this paper, we propose a mechanism to mimic the process of visual priming in deep neural networks in the context of object detection and segmentation. The mechanism transforms an external cue about the presence of a certain class in an image (e.g., "person") to a modulatory signal that affects all layers of the network. This modulatory effect is shown via experimentation to significantly improve object detection performance when the cue is present, more so than a baseline which simply applies post-processing to the network's result. Furthermore, we show that priming early visual layers has a greater effect that doing so for deeper layers. Moreover, the effects of priming are shown to be much more pronounced in difficult images such as very noisy ones.

The remainder of the paper is organized as follows: in Sec. 2 we go over related work from computers vision, psychology and neurophysiology. In Sec. 3 we go over the details of the proposed method. In Sec. 4 we elaborate on various experiments where we evaluate the proposed method in scenarios of object detection and segmentation. We finish with some concluding remarks.

2. Related Work

Context has been very broadly studied in cognitive neuroscience [4, 3, 23, 37, 38, 24, 16] and in computer vision

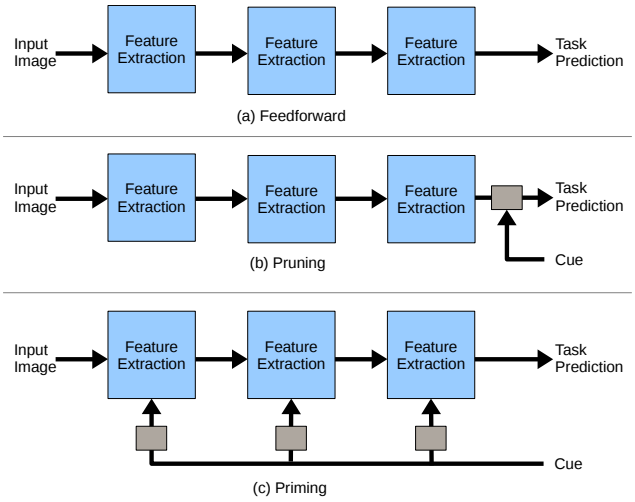


Figure 2: A neural network can be applied to an input in an either unmodified manner (*top*), pruning the results after running (*middle*) or *priming* the network via an external signal (cue) in image to affect all layers of processing (*bottom*).

[12, 10, 34, 33, 27, 39, 22]. It is widely agreed [30] that context plays crucial role for various visual tasks. Attempts have been made to express a tangible definition for context due to the increased use in the computer vision community [34, 33] .

Biederman et al. [4] hypothesizes object-environments dependencies into five categories: probability, interposition, support, familiar size, position. Combinations of some of these categories would form a source of contextual information for tasks such as object detection [33, 30], semantic segmentation [14], and pose estimation [6]. Context consequently is the set of sources that partially or collectively influence the perception of a scene or the objects within [32].

Visual cues originated from contextual sources, depending on the scope they influence, further direct visual tasks at either global or local level [34, 33]. Global context such as scene configuration, imaging conditions, and temporal continuity refers to cues abstracted across the whole scene. On the other hand, local context such as semantic relationships and local-surroundings characterize associations among various parts of similar scenes.

Having delineated various contextual sources, the general process by which the visual hierarchy is modulated prior to a particular task is referred to as visual priming [35, 26]. A cue could be provided either implicitly by a contextual source or explicitly through other modalities such as language.

There has been a tremendous amount of work on using some form of top-down feedback to contextually prime the underlying visual representation for various tasks [37, 38,

24, 16]. The objective is to have signals generated from some task such that they could prepare the visual hierarchy oriented for the primary task. [30] proposes contextual priming and feedback for object detection using the Faster R-CNN framework [29]. The intuition is to modify the detection framework to be able to generate semantic segmentation predictions in one stage. In the second stage, the segmentation primes both the object proposal and classification modules.

Instead of relying on the same modality for the source of priming, [9, 25] proposes to modulate features of a visual hierarchy using the embeddings of the language model trained on the task of visual question answering [1, 17]. In other words, using feature-wise affine transformations, [25] multiplicatively and additively modulates hidden activities of the visual hierarchy using the top-down priming signals generated from the language model, while [30] append directly the semantic segmentation predictions to the visual hierarchy. Recently, [14] proposes to modulate convolutional weight parameters of a neural networks using segmentation-aware masks. In this regime, the weight parameters of the model are directly approached for the purpose of priming.

Although all these methods modulate the visual representation, none has specifically studied the explicit role of category cues to prime the visual hierarchy for object detection and segmentation. In this work, we strive to introduce a consistent parametric mechanism into the neural network framework. The proposed method allows every portion of the visual hierarchy to be primed for tasks such as object detection and semantic segmentation. It should be noted that this use of priming was defined as part of the Selective Tuning (ST) model of visual attention [36]. Other aspects of ST have recently appeared as part of classification and localization networks as well [5, 41], and our work explores yet another dimension of the ST theory.

3. Approach

Assume that we have some network N to perform a task such as object detection or segmentation on an image I. In addition, we are given some cue $h \in \mathcal{R}^{\mathrm{n}}$ about the content of the image. We next describe pruning and priming, how they are applied and how priming is learned. We assume that h is a binary encoding of them presence of some target(s) (e.g, objects) - though this can be generalized to other types of information. For instance, an explicit specification of color, location, orientation, etc, or an encoded features representation as can be produced by a vision or language model. Essentially, one can either ignore this cue, use it to post-process the results, or use it to affect the computation. These three strategies are presented graphically in Fig. 2.

Pruning. In pruning, N is fed an image and we use h to post-process the result. In object detection, all bounding boxes output by N whose class is different than indicated by h are discarded. For segmentation, assume N outputs a score map of size $C \times h \times w$, where L is the number of classes learned by the network, including a background class. We propose two methods of pruning, with complementary effects. The first type increases recall by ranking the target class higher: for each pixel (x,y), we set the value of all score maps inconsistent with h to be $-\infty$, except that of the background. This allows whatever detection of the hinted class to be ranked higher than other which previously masked it. The second type simply sets each pixels which was not assigned by the segmentation the target class to the background class. This decreases recall but increases the precision. These types of pruning are demonstrated in Fig. 8 and discussed below.

Priming. Our approach is applicable to any network N with a convolutional structure, such as a modern network for object detection, e.g. [20]. To enable priming, we freeze all weights in N and add a parallel branch N_p. The role of N_p is to transform an external cue $h \in \mathcal{R}^{\mathrm{n}}$ to modulatory signals which affect all or some of the layers of N. Namely, let L_i be some layer of N. Denote the output of L_i by $x_i \in \mathcal{R}^{c_i \times h_i \times w_i}$ where c_i is the number of feature planes and h_i, w_i are the height and width of the feature planes. Denote the j_{th} feature plane of x_i by $x_{ij} \in \mathcal{R}^{h_i \times w_i}$.

N_p modulates each feature plane x_{ij} by applying to the

$$f_{ij}(x_{ij}, h) = \hat{x}_{ij} \tag{4}$$

The function f_{ij} always operates in a spatially-invariant manner - for each element in a feature plane, the same function is applied. Specifically, we use a simple residual function, that is

$$\hat{x}_{ij} = \alpha_{ij} \cdot x_{ij} + x_{ij} \tag{5}$$

Where the coefficients $\boldsymbol{\alpha_i} = [\alpha_{i1}, \ldots, \alpha_{ic_i}]^T$ are determined by a linear transformation of the cue:

$$\alpha_i = W_i * h \tag{6}$$

An overall view of the proposed method is presented in Fig. 3.

Types of Modulation The modulation in eq. 5 simply adds a calculated value to the feature plane. We have experimented with other types of modulation, namely non-residual ones (e.g, purely multiplicative), as well as following the modulated features with a non-linearity (ReLU), or adding a bias term in addition to the multiplicative part. The single most important dominant ingredient to reach good performance was the residual formulation - without

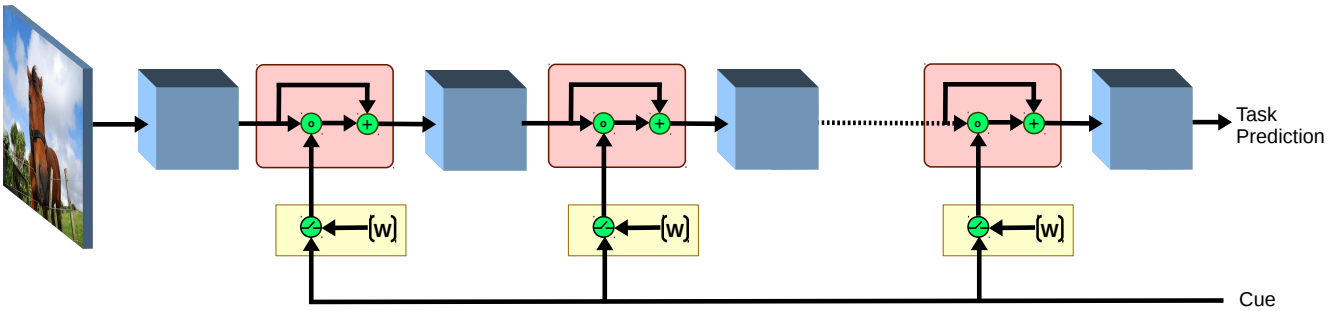


Figure 3: Overall view of the proposed method to prime deep neural networks. A cue about the target in the image is given by task direction, by scene knowledge, or some other source, communicated to each layer via top-down feedback. The process of priming involves affecting each layer of computation of the network by modulating representations along the path.

it, training converged to very poor results. The formulation in eq. 5 performed best without any of the above listed modifications. We note that an additive model, while having converged to better results, is not fully consistent with biologically plausible models ([36]) which involve suppression/selection of visual features, however, it may be considered a first approximation.

Types of Cues The simplest form of a cue h is an indicator vector of the object(s) to be detected, i.e, a vector of 20 zeros and 1 in the coordinate corresponding to "horse", assuming there are 20 possible object classes, such as in Pascal [11]. We call this a *categorical* cue because it explicitly carries semantic information about the object. This means that when a single class k is indicated, α_i becomes the k_{th} column of W_i.

3.1. Training

To learn how to utilize the cue, we freeze the parameters of our original network N and add the network block N_p. During training, with each training example (I_i, y_i) fed into N we feed h_i into N_p, where I_i is an image, y_i is the ground-truth set of bounding boxes and h_i is the corresponding cue. The output and loss functions of the detection network remain the same, and the error is propagated through the parameters of N_p. Fig. 3 illustrates the network. N_p is very lightweight with respect to N, as it only contains parameters to transform from the size of the cue h to at most $K = \sum_i k_i$ where k_i is the number of output feature planes in each layer of the network.

Multiple Cues Per Image. Contemporary object detection and segmentation benchmarks [19, 11] often contain more than one object type per image. In this case, we may set each coordinate in h to 1 iff the corresponding class is present in the image. However, this tends to prevent N_p from learning to modulate the representation of N in a way which allows it to suppress irrelevant objects. Instead, if an image contains k distinct object classes, we duplicate the training sample k times and for each duplicate set the ground truth to contain only one of the classes. This comes at the expense of a longer training time, depending on the average number k over the dataset.

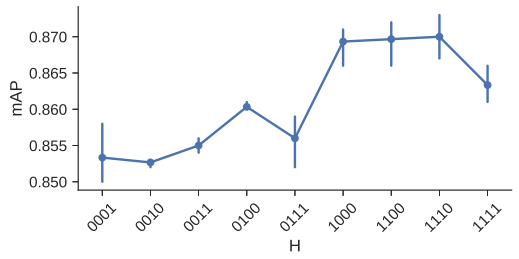
4. Experiments

We evaluate our method on two tasks: object detection and object class segmentation. In each case, we take a pre-trained deep neural network and explore how it is affected by priming or pruning. Our goal here is not necessarily to improve state-of-the-art results but rather to show how usage of top-down cues can enhance performance. Our setting is therefore different than standard object-detection/segmentation scenarios: we assume that some cue about the objects in the scene is given to the network and the goal is to find how it can be utilized optimally. Such information can be either deduced from the scene, such as in contextual priming [30, 18] or given by an external source, or even be inferred from the task, such as in question answering [1, 17].

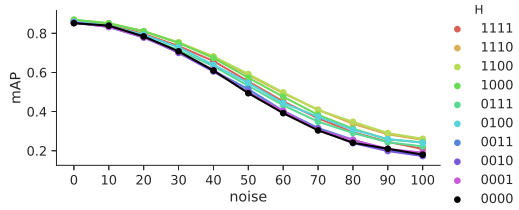
Our experiments are conducted on the Pascal VOC [11] 2007 and 2012 datasets. For priming object detection networks we use pre-trained models of SSD [20] and yolo-v2 [28] and for segmentation we use the FCN-8 segmentation network of [21] and the DeepLab network of [7]. We use the YellowFin optimizer [40] in all of our experiments, with a learning rate of either 0.1 or 0.01 (depending on the task).

4.1. Object Detection

We begin by testing our method on object detection. Using an implementation of SSD [20], we apply a pre-trained detector trained on the trainval sets of Pascal 2012+2007 to the test set of Pascal 2007. We use the SSD-300 variant as described in the paper. In this experiment, we trained and tested on what we cal PAS$^{\#}$: this is a reduced version of Pascal-2007 containing only images with a single object class (but possibly multiple instances). We use this reduced dataset to test various aspects of our method, as



(a)



(b)

Figure 4: (a) Performance gains by priming different parts of the SSD objects detector. Priming early parts of the network causes the most significant boost in performance. black dashed line shows performance by pruning. (b) Testing variants of priming against increasing image noise. The benefits of priming become more apparent in difficult viewing conditions. The x axis indicates which block of the network was primed (1 for primed, 0 for not primed).

detailed in the following subsections. Without modification, the detector attains a mAP (mean-average precision) of 81.4% on PAS#(77.4% on the full test set of Pascal 2007). Using simple pruning as described above, this increases to 85.2%. This large boost in performance is perhaps not surprising, since pruning effectively removes all detections of classes that do not appear in the image. The remaining errors are those of false alarms of the "correct" class or misdetections.

Deep vs Shallow Priming

We proceed to the main result, that is, how priming affects detection. The SSD object detector contains four major components: (1) a pre-trained part made up of some of the layers of vgg-16 [31] (a.k.a the "base network" in the SSD paper), (2) some extra convolutional layers on top of the vgg-part, (3) a localization part and (4) a class confidence part. We name these part *vgg*, *extra*, *loc* and *conf* respectively.

To check where priming has the most significant impact, we select different subsets of these components and denote them by 4-bit binary vectors $s_i \in \{0,1\}^4$, where the bits correspond from left to right to the vgg,extra,localization and confidence parts. For example, $s = 1000$ means letting N_p affect only the earliest (*vgg*) part of the detector, while all other parts remain unchanged by the priming (except

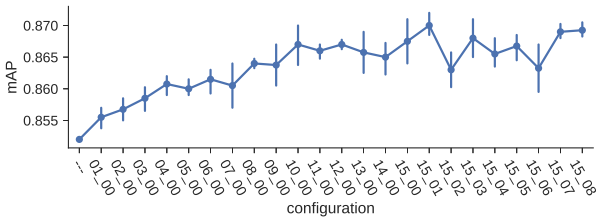


Figure 5: Effects of early priming: we show how mAP increases when we allow priming to affect each time another layer, from the very bottom of the network. Priming early layers has a more significant effect than doing so for deeper ones. The numbers indicate how many layers were primed from the first,second blocks of the SSD network, respectively.

indirectly affecting the deeper parts of the net). We train N_p on 10 different configurations: these include priming from the deepest layers to the earliest: 1111, 0111, 0011, 0001 and from the earliest layer to the deepest: 1000, 1100, 1110. We add 0100 and 0010 to check the effect of exclusive control over middle layers and finally 0000 as the default configuration in which N_p is degenerate and the result is identical to pruning. Fig 4 (a) shows the effect of priming each of these subsets of layers on PAS#. Priming early layers (those at the bottom of the network) has a much more pronounced effect than priming deep layers. The single largest gain by priming a single component is for the *vgg* part: 1000 boosts performance from 85% to 87.1%. A smaller gain is attained by the *extra* component: 86.1% for 0100. The performance peaks at **87.3**% for 1110, though this is only marginally higher than attained by 1100 - priming only the first two parts.

Ablation Study

Priming the earliest layers (*vgg*+*extra*) of the SSD object detector brings the most significant boost in performance. The first component described above contains 15 convolutional layers and the second contains 8 layers, an overall total of 23. To see how much we can gain with priming on the first few layers, we checked the performance on PAS# when training on the first k layers only, for each $k \in \{1, 2, \ldots 23\}$. Each configuration was trained for 4000 iterations. Fig. 5 shows the performance obtained by each of these configurations, where i_j in the x-axis refers to having trained the first i layers and the first j layers of the first and second parts respectively. We see that the very first convolutional layer already boosts performance when primed. The improvement continues steadily as we add more layers and fluctuates around 87% after the 15th layer. The fluctuation is likely due to randomness in the training process. This further shows that priming has strong effects when applied

to very early layers of the network.

Detection in Challenging Images

As implied by the introduction, perhaps one of the cases where the effect of priming is stronger is when facing a challenging image, such as adverse imaging conditions, low lighting, camouflage, noise. As one way to test this, we compared how priming performs under noise. We took each image in the test set of Pascal 2007 and added random Gaussian noise chosen from a range of standard deviations, from 0 to 100 in increments of 10. The noisy test set of PAS$^{\#}$ with variance σ is denoted PAS$^{\#}{}_{\mathrm{N}(\sigma)}$. For each σ, we measure the mAP score attained by either pruning or priming. Note that none of our experiments involved training with only images - these are only used for testing. We plot the results in Fig. 4 (b). As expected, both methods suffer from decreasing accuracy as the noise increases. However, priming is more robust to increasing levels of noise; the difference between the two methods peaks at a moderate level of noise, that is, $\sigma = 80$, with an advantage of $\mathbf{10.7}\%$ in mAP: 34.8% compared to 24.1% by pruning. The gap decreases gradually to 6.1% (26.1% vs 20%) for a noise level of $\sigma = 100$. We believe that this is due to the early-layer effects of priming on the network, selecting features from the bottom up to match the cue. Fig 6 shows qualitative examples, comparing priming versus pruning: we increase the noise from top to bottom and decrease the threshold (increase the sensitivity) from left to right. We show in each image only the top few detections of each method to avoid clutter. Priming allows the detector to find objects in images with high levels of noise (see lower rows of a,b). In some cases priming proves to be *essential* for the detection: lowering the un-primed detector's threshold to a minimal level does not increase the recall of the desired object (a, 4th row); in fact, it only increases the number of false alarms (b, 2nd row, last column). Priming, on the other hand, is often less sensitive to a low threshold and the resulting detection persists along a range thereof.

4.2. Cue Aware Training

In this section, we also test priming on an object detection task as well as segmentation with an added ingredient - multi-cue training and testing. In Sec. 4.1 we limited ourselves to the case where there is only one object class per image. This limitation is often unrealistic. To allow multiple priming cues per image, we modify the training process as follows: for each training sample $< I, gt >$ containing object classes $c_1, \ldots c_k$ we split the training example for I to k different tuples $< I_i, h_i, gt_i >, i \in \{1 \ldots k\}$, where I_i are all identical to I, h_i indicate the presence of class c_i and gt_i is the ground-truth gt reduced to contain only the objects of class c_i - meaning the bounding boxes for detection, or the masks for segmentation. This explicitly coerces the priming network N_p to learn how to force the output to correspond to the given cue, as the input image remains the same but the cue and desired output change together. We refer to this method multi-cue aware training (CAT for short) , and refer to the unchanged training scheme as regular training.

Multi-Cue Segmentation

Here, we test the multi-cue training method on object class segmentation. We begin with the FCN-8 segmentation network of [21]. We train on the training split of SBD (Berkeley Semantic Boundaries Dataset and Benchmark) dataset [13], as is done in [42, 7, 8, 21]. We base our code on an unofficial PyTorch[2] implementation[3]. Testing is done of the validation set of Pascal 2011, taking care to avoid overlapping images between the training set defined by [13] [4], which leaves us with 736 validation images. The baseline results average IOU score of 65.3%. As before, we let the cue be a binary encoding of the classes present in the image. We train and test the network in two different modes: one is by setting for each training sample (and testing) the cue so $h_i = 1$ if the current image contains at least one instance of class i and 0 otherwise. The other is the multi-cue method we describe earlier, i.e , splitting each sample to several cues with corresponding ground-truths so each cue is a one-hot encoding, indicating only a single class. For both training strategies, testing the network with a cue creates a similar improvement in performance, from 65.3% to 69% for regular training and to 69.2% for multi-cue training.

The main advantage of the multi-cue training is that it allows the priming network N_p to force N to focus on different objects in the image. This is illustrated in Fig. 7. The top row of the figure shows from left to right an input image and the resulting segmentation masks when the network is cued with classes *bottle*, *diningtable* and *person*. The bottom row is cued with *bus*, *car*, *person*. The cue-aware training allows the priming network to learn how to suppress signals relating to irrelevant classes while retaining the correct class from the bottom-up.

Types of Pruning. As mentioned in Sec. 3, we examine two types of pruning to post-process segmentation results. One type removes image regions which were wrongly labeled as the target class, replacing them with background and the other increases the recall of previously missed segmentation regions by removing all classes except the target class and retaining pixels where the target class scored higher than the background. The first type increases precision but cannot increase recall. The second type increases recall but possibly hinders precision. We found that both

[2] http://pytorch.org/

[3] https://github.com/wkentaro/pytorch-fcn

[4] for details, please refer to https://github.com/shelhamer/fcn.berkeleyvision.org/tree/master/data/pascal

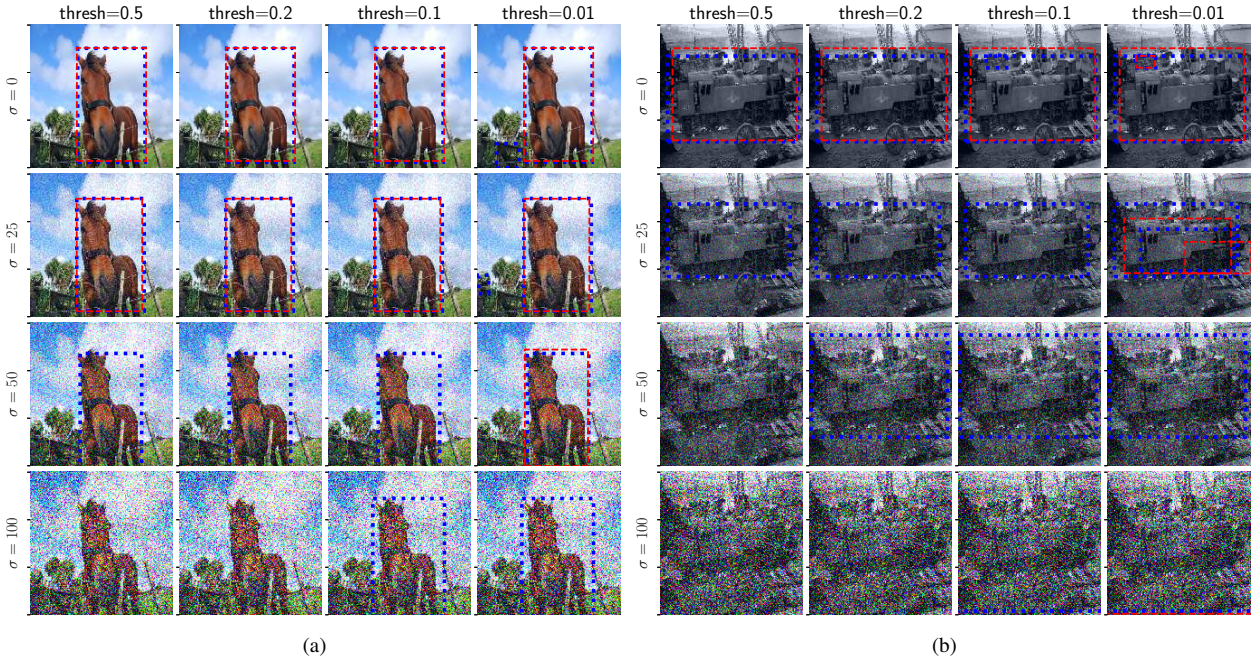
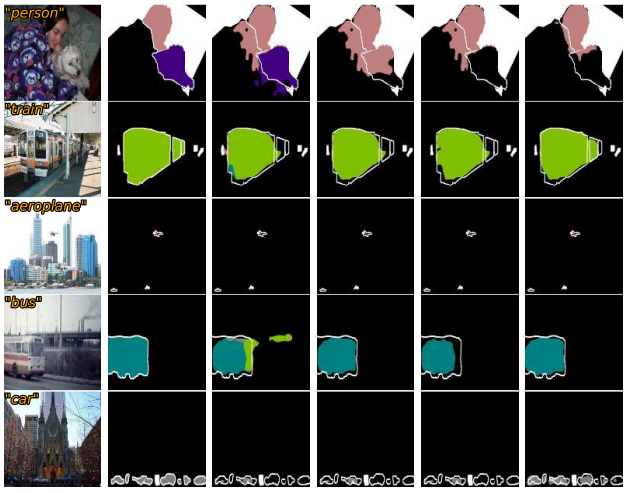


Figure 6: Priming vs. Pruning. Priming a detector allows it to find objects in images with high levels of noise while mostly avoiding false-alarms. Left to right (*a,b*): decreasing detection thresholds (increasing sensitivity). Top to bottom: increasing levels of noise. Priming (**blue** dashed boxes) is able to detect the horse (*a*) across all levels of noise, while pruning (**red** dashed boxes) does not. For the highest noise level, the original classifier does not detect the horse at all - so pruning is ineffective. (*b*) Priming enables detection of the train for all but the most severe level of noise. Decreasing the threshold for pruning only produces false alarms. We recommend viewing this figure in color on-line.

Figure 7: Effect of priming a segmentation network with different cues. In each row, we see an input image and the output of the network when given different cues. Top row: cues are resp. bottle, diningtable, person. Given a cue (e.g, *bottle*), the network becomes more sensitive to bottle-like image structures while suppressing others. This happens not by discarding results but rather by affecting computation starting from the early layers.

types results in a similar overall mean-IOU. Figure 8 shows some examples where both types of pruning result in segmentations inferior to the one resulting by priming: post-processing can increase recall by lowering precision (first row, column d) or increase precision by avoiding false-detections (second and fourth row, column e), priming (column f) increases both recall and precision. The second, and fourth rows missing parts of the train/bus are recovered while removing false classes. The third and fifth rows previously undetected small objects are now detected. The person (first row) is segmented more accurately.

DeepLab. Next, we use the DeepLab [7] network for semantic-segmentation with ResNet-101 [15] as a base network. We do not employ a CRF as post-processing. The mean-IOU of the baseline is 76.3%. Using Priming, increases this to 77.15%. While in this case priming does not improve as much as in the other cases we tested, we find that it is especially effective at enabling the network to discover small objects which were not previously segmented by the non-primed version: the primed network discovers 57 objects which were not discovered by the unprimed network, whereas the latter discovers only 3 which were not discovered by the former. Fig. 9 shows some representative examples of where priming was advantageous. Note how the bus, person, (first three rows) are segmented by the primed network (last column). We hypothesize that the



(a) input (b) gt (c) regular (d) prune-2 (e) prune-1 (f) priming

Figure 8: Comparing different methods of using a cue to improve segmentation: From left to right: input image (with cue overlayed), ground-truth (all classes), unprimed segmentation, pruning type-2, pruning type-1, and priming. In each image, we aid the segmentation network by adding a cue (e.g, "plane"). White regions are marked as "don't care" in the ground truth.

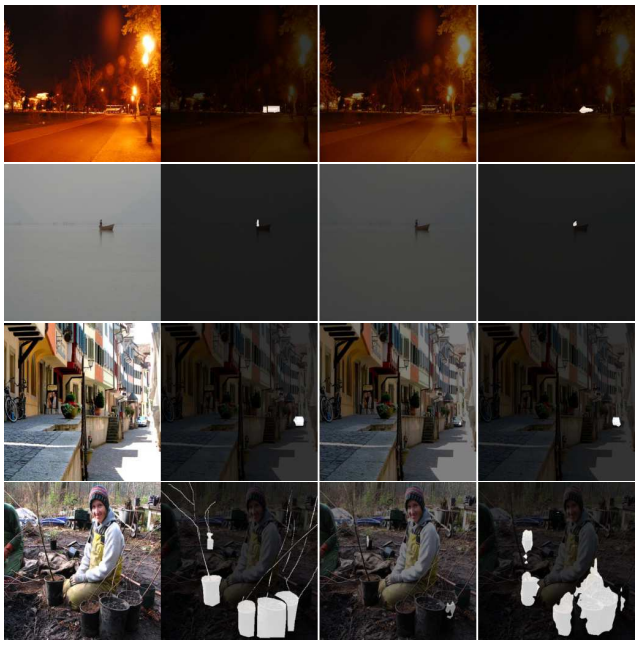


Figure 9: Priming a network allows discovery of small objects which are completely missed by the baseline, or ones with uncommon appearance (last row). From left to right: input image, ground-truth, baseline segmentation [7], primed network.

priming process helps increase the sensitivity of the network to features relevant to the target object. The last row shows a successful segmentation of potted plants with a rather atypical appearance.

Multi-Cue Object Detection

We apply the CAT method to train priming on object detection as well. For this experiment, we use the YOLOv2 method of [28]. The base network we used is a port of the original network, known as YOLOv2 544 × 544. Trained on the union of Pascal 2007 and 2012 datasets, it is reported by the authors to obtain 78.6% mAP on the test set of Pascal 2007. The implementation we use[5] reaches a slightly lower 76.8%, with a PyTorch port of the network weights released by the authors. We use all the convolutional layers of DarkNet (the base network of YOLOv2) to perform priming. We freeze all network parameters of the original detection network and train a priming network with the multi-cue training method for 25 epochs. When using only pruning, performance on the test-set improves to 78.2% mAP. When we include priming as well, this goes up to 80.6%,

5. Conclusion

We have presented a simple mechanism to prime neural networks, as inspired by psychological top-down effects known to exist in human observers. We have tested the proposed method on two tasks, namely object detection and segmentation, using two methods for each task, and comparing it to simple post-processing of the output. Our experiments confirm that as is observed in humans, effective usage of a top-down signal to modulate computations from early layers not only improves robustness to noise but also facilitates better object detection and segmentation, enabling detection of objects which are missed by the baselines without compromising precision, notably so for small objects and those having an atypical appearance or in challenging, noisy images, without having trained for such images in advance. In the future we intend to explore additional priming mechanisms and richer types of cues.

Acknowledgement This research was supported by several sources, via grants to the senior author, for which the authors are grateful: Air Force Office of Scientific Research USA (FA9550-18-1-0054), the Canada Research Chairs Program (950-231659), and the Natural Sciences and Engineering Research Council of Canada (RGPIN-2016-05352).

References

[1] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In

[5] https://github.com/marvis/pytorch-yolo2

Proceedings of the IEEE International Conference on Computer Vision, pages 2425–2433, 2015. 2, 4

[2] Mandy V Bartsch, Kristian Loewe, Christian Merkel, Hans-Jochen Heinze, Mircea A Schoenfeld, John K Tsotsos, and Jens-Max Hopf. Attention to color sharpens neural population tuning via feedback processing in the human visual cortex hierarchy. *Journal of Neuroscience*, 37(43):10346–10357, 2017. 1

[3] Irving Biederman. *On the semantics of a glance at a scene*. 1981. 2

[4] Irving Biederman, Robert J Mezzanotte, and Jan C Rabinowitz. Scene perception: Detecting and judging objects undergoing relational violations. *Cognitive psychology*, 14(2):143–177, 1982. 2

[5] Mahdi Biparva and John Tsotsos. STNet: Selective Tuning of Convolutional Networks for Object Localization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[6] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4733–4742, 2016. 2

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 4, 4.2.1, 4.2.1, 9

[8] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1635–1643, 2015. 4.2.1

[9] Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. *arXiv preprint arXiv:1707.00683*, 2017. 2

[10] Santosh K Divvala, Derek Hoiem, James H Hays, Alexei A Efros, and Martial Hebert. An empirical study of context in object detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1271–1278. IEEE, 2009. 2

[11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 3, 3.1, 4

[12] Carolina Galleguillos and Serge Belongie. Context based object categorization: A critical survey. *Computer vision and image understanding*, 114(6):712–722, 2010. 2

[13] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 991–998. IEEE, 2011. 4.2.1

[14] Adam W Harley, Konstantinos G Derpanis, and Iasonas Kokkinos. Segmentation-Aware Convolutional Networks Using Local Attention Masks. *arXiv preprint arXiv:1708.04607*, 2017. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4.2.1

[16] Andrew Hollingworth. Does consistent scene context facilitate object perception? *Journal of Experimental Psychology: General*, 127(4):398, 1998. 2

[17] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *arXiv preprint arXiv:1612.06890*, 2016. 2, 4

[18] Harish Katti, Marius V Peelen, and SP Arun. Object detection can be improved using human-derived contextual expectations. *arXiv preprint arXiv:1611.07218*, 2016. 4

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 3.1

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 3, 4, 4.1

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 4, 4.2.1

[22] Kevin P Murphy, Antonio Torralba, and William T Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *Advances in neural information processing systems*, pages 1499–1506, 2004. 2

[23] Aude Oliva and Antonio Torralba. The role of context in object recognition. *Trends in cognitive sciences*, 11(12):520–527, 2007. 2

[24] tephen E Palmer. The effects of contextual scenes on the identification of objects. *Memory & Cognition*, 3(5):519–526, 1975. 2

[25] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer. *arXiv preprint arXiv:1709.07871*, 2017. 2

[26] Michael I Posner, Mary Jo Nissen, and William C Ogden. Attended and unattended processing modes: The role of set for spatial location. *Modes of perceiving and processing information*, 137:158, 1978. 2

[27] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on*, pages 1–8. IEEE, 2007. 2

[28] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. *arXiv preprint arXiv:1612.08242*, 2016. 4, 4.2.2

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian 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

[30] Abhinav Shrivastava and Abhinav Gupta. Contextual priming and feedback for faster r-cnn. In *European Conference on Computer Vision*, pages 330–348. Springer, 2016. 2, 4

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4.1.1

[32] Thomas M Strat. Employing contextual information in computer vision. *DARPA93*, pages 217–229, 1993. 2

[33] Antonio Torralba. Contextual priming for object detection. *International journal of computer vision*, 53(2):169–191, 2003. 2

[34] Antonio Torralba, Kevin P Murphy, William T Freeman, and Mark A Rubin. Context-based vision system for place and object recognition. In *null*, page 273. IEEE, 2003. 2

[35] John K Tsotsos. *A computational perspective on visual attention*. MIT Press, 2011. 1, 2

[36] John K. Tsotsos, Scan M. Culhane, Winky Yan Kei Wai, Yuzhong Lai, Neal Davis, and Fernando Nuflo. Modeling visual attention via selective tuning. *Artificial Intelligence*, 78(1–2):507–545, 1995. Special Volume on Computer Vision. 2, 3

[37] Endel Tulving, Daniel L Schacter, et al. Priming and human memory systems. *Science*, 247(4940):301–306, 1990. 2

[38] Gagan S Wig, Scott T Grafton, Kathryn E Demos, and William M Kelley. Reductions in neural activity underlie behavioral components of repetition priming. *Nature neuroscience*, 8(9):1228–1233, 2005. 2

[39] Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 702–709. IEEE, 2012. 2

[40] Jian Zhang, Ioannis Mitliagkas, and Christopher Ré. YellowFin and the Art of Momentum Tuning. *arXiv preprint arXiv:1706.03471*, 2017. 4

[41] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down Neural Attention by Excitation Backprop. In *European Conference on Computer Vision*, pages 543–559. Springer, 2016. 2

[42] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *arXiv preprint arXiv:1612.01105*, 2016. 4.2.1