

# Beyond Skip Connections: Top-Down Modulation for Object Detection

Abhinav Shrivastava<sup>1,3</sup>Rahul Sukthankar<sup>3</sup>Jitendra Malik<sup>2,3</sup>Abhinav Gupta<sup>1,3</sup><sup>1</sup>Carnegie Mellon University    <sup>2</sup>University of California, Berkeley    <sup>3</sup>Google Research

## Abstract

In recent years, we have seen tremendous progress in the field of object detection. Most of the recent improvements have been achieved by targeting deeper feedforward networks. However, many hard object categories such as bottle, remote, etc. require representation of fine details and not just coarse, semantic representations. But most of these fine details are lost in the early convolutional layers. What we need is a way to incorporate finer details from lower layers into the detection architecture. Skip connections have been proposed to combine high-level and low-level features, but we argue that selecting the right features from low-level requires top-down contextual information. Inspired by the human visual pathway, in this paper we propose top-down modulations as a way to incorporate fine details into the detection framework. Our approach supplements the standard bottom-up, feedforward ConvNet with a top-down modulation (TDM) network, connected using lateral connections. These connections are responsible for the modulation of lower layer filters, and the top-down network handles the selection and integration of contextual information and low-level features. The proposed TDM architecture provides a significant boost on the COCO benchmark, achieving 28.6 AP for VGG16 and 35.2 AP for ResNet101 networks. Using InceptionResNetv2, our TDM model achieves 37.3 AP, which is the best single-model performance to-date on the COCO testdev benchmark, without any bells and whistles.

## 1. Introduction

Convolutional neural networks (ConvNets) have revolutionized the field of object detection [4, 14, 16, 17, 41, 45, 49]. Most standard ConvNets are bottom-up, feed-forward architectures constructed using repeated convolutional layers (with non-linearities) and pooling operations [24, 28, 46–48]. These convolutional layers learn invariances and the spatial pooling increases the receptive field of subsequent layers; thus resulting in a coarse, highly semantic representation at the final layer.

However, consider the images shown in Figure 1. Detecting and recognizing an object like the bottle in the left



Figure 1. Detecting objects such as the bottle or remote shown above requires low-level finer details as well as high-level contextual information. In this paper, we propose a top-down modulation (TDM) network, which can be used with any bottom-up, feedforward ConvNet. We show that the features learnt by our approach lead to significantly improved object detection.

image or remote in the right image requires extraction of very fine details such as the vertical and horizontal parallel edges. But these are exactly the type of edges ConvNets try to gain invariance against in early convolutional layers. One can argue that ConvNets can learn not to ignore such edges when in context of other objects like table. However, objects such as table do not emerge until very late in feed-forward architecture. So, how can we incorporate these fine details in object detection?

A popular solution is to use variants of ‘skip’ connections [4, 10, 22, 35, 43, 53], that capture these finer details from lower convolutional layers with *local* receptive fields. But simply incorporating high-dimensional skip features into detection does not yield significant improvements due to overfitting caused by curse of dimensionality. What we need is a selection/attention mechanism that selects the relevant features from lower convolutional layers.

We believe the answer lies in the process of *top-down modulation*. In the human visual pathway, once receptive field properties are tuned using feedforward processing, *top-down modulations* are evoked by feedback and horizontal connections [27, 30]. These connections modulate representations at multiple levels [13, 15, 37, 54, 55] and are responsible for their selective combination [7, 25]. We argue that the use of skip connections is a special case of this process, where the *modulation* is relegated to the final classifier, which directly tries to influence lower layer features and/or learn how to combine them.

In this paper, we propose to incorporate the *top-down modulation* process in the ConvNet itself. Our approach supplements the standard bottom-up, feedforward ConvNet with a top-down network, connected using lateral connections. These connections are responsible for the *modulation* and *selection* of the lower layer filters, and the top-down network handles the *integration* of features.

Specifically, after a bottom-up ConvNet pass, the final high-level semantic features are transmitted back by the top-down network. Bottom-up features at intermediate depths, after lateral processing, are combined with the top-down features, and this combination is further transmitted down by the top-down network. Capacity of the new representation is determined by lateral and top-down connections, and optionally, the top-down connections can increase the spatial resolution of features. These final, possibly high-res, top-down features inherently have a combination of *local* and *larger* receptive fields.

The proposed Top-Down Modulation (TDM) network is trained end-to-end and can be readily applied to any base ConvNet architecture (*e.g.*, VGG [46], ResNet [24], Inception-Resnet [47] *etc.*). To demonstrate its effectiveness, we use the proposed network in the standard Faster R-CNN detection method [41] and evaluate on the challenging COCO benchmark [33]. We report a consistent and significant boost in performance on all metrics across network architectures. TDM network increases the performance of vanilla Faster R-CNN with: (a) VGG16 from 23.3 AP to textbf{28.6} AP, (b) ResNet101 from 31.5 AP to **35.2** AP, and (c) InceptionResNetv2 from 34.7 AP to **37.2** AP. These are the best performances reported to-date for these architectures without any bells and whistles (*e.g.*, multi-scale features, iterative box-refinement). Furthermore, we see drastic improvements in small objects (*e.g.*, +**4.5** AP) and in objects where selection of fine details using top-down context is important.

## 2. Related Work

After the resurgence of ConvNets for image classification [8, 28], they have been successfully adopted for a variety of computer vision tasks such as object detection [16, 17, 41, 49], semantic segmentation [3, 6, 34, 35], instance segmentation [21, 22, 38], pose estimation [50, 51], depth estimation [9, 52], edge detection [53], optical flow predictions [11, 40] *etc.* However, by construction, final ConvNet features lack the finer details that are captured by lower convolutional layers. These finer details are considered necessary for a variety of recognition tasks, such as accurate object localization and segmentation.

To counter this, ‘skip’ connections have been widely used with ConvNets. Though the specifics of methods vary widely, the underlying principle is same: using or combining finer features from lower layers and coarse seman-

tic features for higher layers. For example, [4, 10, 22, 43] combine features from multiple layers for the final classifier; while [4, 43] use subsampled features from finer scales, [10, 22] upsample the features to the finest scale and use their combination. Instead of combining features, [34, 35, 53] do independent predictions at multiple layers and average the results. In our proposed framework, such upsampling, subsampling and fusion operations can be easily controlled by the lateral and top-down connections.

The proposed TDM network is conceptually similar to the strategies explored in other contemporary works [3, 32, 39, 40, 42]. All methods, including ours, propose architectures that go beyond the standard skip-connection paradigm and/or follow a coarse-to-fine strategy when using features from multiple levels of the bottom-up feature hierarchy. However, different methods focus on different tasks which guide their architectural design and training methodology.

Conv-deconv [36] and encoder-decoder style networks [3, 42] have been used for image segmentation to utilize finer features via lateral connections. These connections generally use the ‘unpool’ operation [56], which merely inverts the spatial pooling operation. Moreover, there is no modulation of bottom-up network. In comparison, our formulation is more generic and is responsible for the flow of high-level context features [37].

Pinheiro *et al.* [39] focus on refining class-agnostic object proposals by first selecting proposals *purely* based on bottom-up feedforward features [38], and then *post-hoc* learning how to refine each proposal independently using top-down and lateral connections (due to the computational complexity, only a few proposals can be selected to be refined). We argue that this use of top-down and lateral connections for refinement is sub-optimal for detection because it relies on the proposals selected based on feedforward features, which are insufficient to represent small and difficult objects. This training methodology also limits the ability to update the feedforward network through lateral connections. In contrast to this, we propose to learn better features for recognition tasks in an end-to-end trained system, and these features are used for both proposal generation and object detection. Similar to the idea of coarse-to-fine refinement, Ranjan and Black [40] propose a coarse-to-fine spatial pyramid network, which computes a low resolution residual optical flow and iteratively improves predictions with finer pyramid levels. This is akin to *only* using a specialized top-down network, which is suited for low-level tasks (like optical flow) but not for recognition tasks. Moreover, such purely coarse-to-fine networks cannot utilize models pre-trained on large-scale datasets [8], which is important for recognition tasks [17]. Therefore, our approach learns representation using bottom-up (fine-to-coarse), top-down (coarse-to-fine) and lateral networks simultaneously, and can use different pre-trained modules.

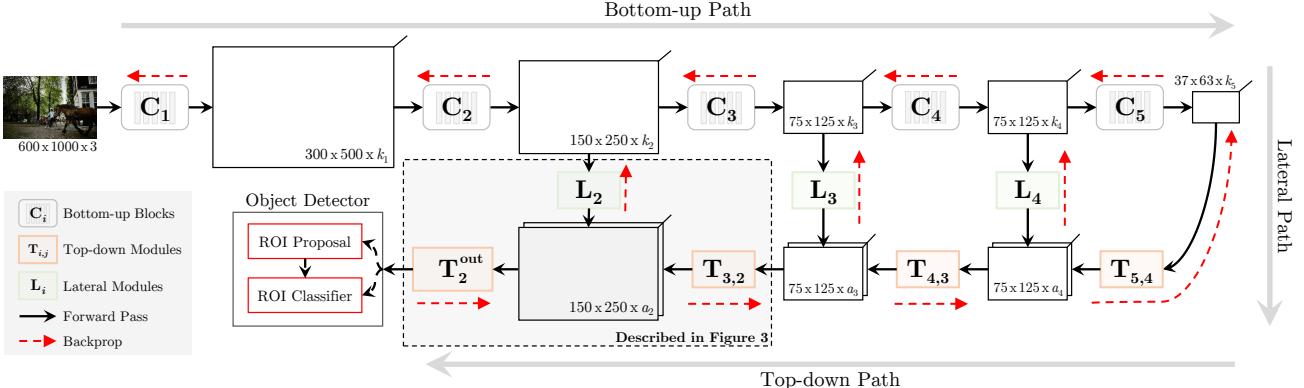


Figure 2. The illustration shows an example of **Top-Down Modulation** (TDM) Network, which is integrated with the bottom-up network with lateral connections.  $C_i$  are bottom-up, feedforward feature blocks,  $L_i$  are the lateral modules which transform low level features for the top-down contextual pathway. Finally,  $T_{j,i}$ , which represent flow of top-down information from index  $j$  to  $i$ . Individual components are explained in Figure 3 and 4.

The proposed top-down network is closest to the recent work of Lin *et al.* [32], developed concurrently to ours, on feature pyramid network for object detection. Lin *et al.* [32] use bottom-up, top-down and lateral connections to learn a feature pyramid, and require multiple proposal generators and region classifiers on each level of the pyramid. In comparison, the proposed top-down modulation focuses on using these connections to learn a single final feature map that is used by a single proposal generator and region classifier.

There is strong evidence of such top-down context, feedback and lateral processing in the human visual pathway [7, 13, 15, 25, 27, 29, 30, 37, 54, 55]; wherein, the top-down signals are responsible for modulating low-level features [13, 15, 37, 54, 55] as well as act as attentional mechanism for selection of features [7, 25]. In this paper, we propose a computation model that captures some of these intuitions and incorporates them in a standard ConvNets, giving substantial performance improvements.

Our top-down framework is also related to the process of contextual feedback [2]. To incorporate top-down feedback loop in ConvNets, contemporary works [5, 12, 31, 44], have used ‘unrolled’ networks (trained stage-wise). The proposed top-down network with lateral connections explores a complementary paradigm and can be readily combined with them. Contextual features have also been used for ConvNets based object detectors; *e.g.*, using other objects [20] or regions [18] as context. We believe the proposed top-down path can naturally transmit these contextual features.

### 3. Top-Down Modulation (TDM)

Our goal is to incorporate *top-down modulation* into current object detection frameworks. The key idea is to select/attend to fine details from lower level feature maps based on top-down contextual features and select top-down contextual features based on the fine low-level details. We formalize this by proposing a simple top-down modulation (TDM) network as shown in Figure 2.

The TDM network starts from the last layer of bottom-up feedforward network. For example, in the case of VGG16, the input to the first layer of the TDM network is the conv5\_3 output. Every layer of TDM network also gets the bottom-up features as inputs via lateral connections. Thus, the TDM network learns to: (a) transmit high-level contextual features that guide the learning and selection of relevant low-level features, and (b) use the bottom-up features to select the contextual information to transmit. The output of the proposed network captures both pertinent finer details and high-level information.

### 3.1. Proposed Architecture

An overview of the proposed framework is illustrated in Figure 2. The standard bottom-up network is represented by blocks of layers, where each block  $C_i$  has multiple operations. The TDM network hinges on two key components: a lateral module  $L$ , and a top-down module  $T$  (see Figure 3). Each lateral module  $L_i$  takes in a bottom-up feature  $x_i^C$  (output of  $C_i$ ) and produces the corresponding lateral feature  $x_i^L$ . These lateral features  $x_i^L$  and top-down features  $x_j^T$  are combined, and optionally upsampled, by the  $T_{j,i}$  module to produce the top-down features  $x_i^T$ . These modules,  $T_{j,i}$  and  $L_i$ , control the capacity of the modulation network by changing their output feature dimensions.

The feature from the last top-down module  $T_i^{\text{out}}$  is used for the task of object detection. For example, in Figure 2, instead of  $x_5^C$ , we use  $T_2^{\text{out}}$  as input to ROI proposal and ROI classifier networks of the Faster R-CNN [41] detection system (discussed in Section 4.1). During training, gradient updates from the object detector backpropagate via top-down and lateral modules to the  $C_i$  blocks. The lateral modules  $L_i$  learn how to transform low-level features and the top-down modules  $T_i$  learn what semantic or context information to preserve in the top-down feature transmission as well as the selection of relevant low-level lateral features. Ultimately, the bottom-up features are modulated to adapt for

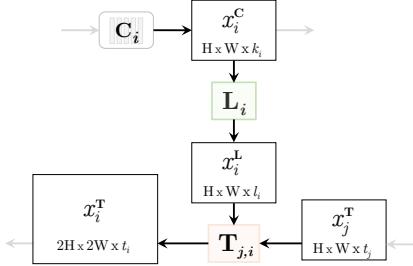


Figure 3. The basic building blocks of Top-Down Modulation Network (detailed Section 3.1).

this new representation.

**Architecture details.** The top-down and lateral modules described above are essentially small ConvNets, which can vary from a single or a hierarchy of convolutional layers to more involved blocks with Residual [24] or Inception [47] units. In this paper, we limit our study by using modules with a single convolutional layer and non-linearity to analyze the impact of top-down modulation.

A detailed example of lateral and top-down modules is illustrated in Figure 4. The lateral module  $L_i$  is a  $3 \times 3$  convolutional layer with ReLU non-linearity, which transforms an  $(H_i \times W_i \times k_i)$  input  $x_i^C$ , to  $(H_i \times W_i \times l_i)$  lateral feature  $x_i^L$ . The top-down module  $T_{j,i}$  is also a  $3 \times 3$  convolutional layer with ReLU, that combines this lateral feature with  $(H_i \times W_i \times t_j)$  top-down feature  $x_j^T$ , to produce an intermediate output  $(H_i \times W_i \times t_i)$ . If the resolution of next lateral feature  $x_{i+1}^L$  is higher than the previous lateral feature (e.g.,  $H_{i+1} = 2 \times H_i$ ), then  $T_{j,i}$  also upsamples the intermediate output to produce  $(H_{i+1} \times W_{i+1} \times t_i)$  top-down feature  $x_j^T$ . In Figure 2, we denote  $a_i = t_j + l_i$  for simplicity. The final  $T_i^{\text{out}}$  module can additionally have a  $1 \times 1$  convolutional layer with ReLU to output a  $(H_i \times W_i \times k_{\text{out}})$  feature, which is used by the detection system.

Varying  $l_i$ ,  $t_i$  and  $k_{\text{out}}$  controls the capacity of the top-down modulation system and dimension of the output features. These hyperparameters are governed by the base network design, detection system and hardware constraints (discussed in Section 4.3). Notice that the upsampling step is optional and depends on the content and arrangement of  $C$  blocks (e.g., no upsampling by  $T_4$  in Figure 2). Also, the first top-down module ( $T_5$  in the illustration) only operates on  $x_5^C$  (the final output of the bottom-up network).

**Training methodology.** Integrating top-down modulation framework into a bottom-up ConvNet is only meaningful when the latter can represent high-level concepts in higher layers. Thus, we typically start with a pre-trained bottom-up network (see Section 6.4 for discussion). Starting with this pre-trained network, we find that progressively building the top-down network performs better in general. Therefore, we add one new pair of lateral and top-down modules at a time. For example, for the illustration in Figure 2, we

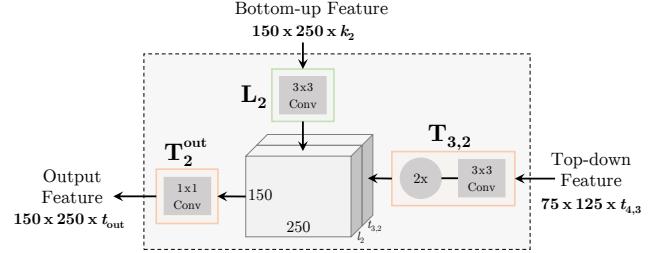


Figure 4. An example with details of top-down modules and lateral connections. Please see Section 3.1 for details of the architecture.

will begin by adding  $(L_4, T_{5,4})$  and use  $T_4^{\text{out}}$  to get features for object detection. After training  $(L_4, T_{5,4})$  modules, we will add the next pair  $(L_3, T_{4,3})$  and use a new  $T_3^{\text{out}}$  module to get features for detection; and we will repeat this process. With each new pair, the entire network, top-down and bottom-up along with lateral connections, is trained end-to-end. Implementation details of this training methodology depends on the base network architecture, and will be discussed in Section 4.3.

To better understand the impact of the proposed TDM network, we conduct extensive experimental evaluation; and provide ablation analysis of various design decisions. We describe our approach in detail (including preliminaries and implementation details) in Section 4 and present our results in Section 5. We also report ablation analysis in Section 6. We would like to highlight that the proposed framework leads to substantial performance gains across different base network architectures, indicating its wide applicability.

## 4. Approach Details

In this section, we describe the preliminaries and provide implementation details of our top-down modulation (TDM) network under various settings. We first give a brief overview of the object detection system and the ConvNet architectures used throughout this paper.

### 4.1. Preliminaries: Faster R-CNN

We use the Faster R-CNN [41] framework as our base object detection system. Faster R-CNN consists of two core modules: 1) ROI Proposal Network (RPN), which takes an image as input and proposes rectangular regions of interests (ROIs); and 2) ROI Classifier Network (RCN), which is a Fast R-CNN [16] detector that classifies these proposed regions and learns to refine ROI coordinates. Given an image, Faster R-CNN first uses a ConvNet to extract features that are shared by both RPN and RCN. RPN uses these features to propose candidate ROIs, which are then classified by RCN. The RCN network projects each ROI onto the shared feature map and performs the ‘ROI Pooling’ [16, 23] operation to extract a fixed length representation. Finally, this feature is used for classification and box regression. See [16, 23, 26, 41] for details.

Table 1. Base networks architecture details for VGG16, ResNet101 and InceptionResNetv2. Legend:  $C_i$ : bottom-up block id, N: number of convolutional filters, NR: number of residual units, NI: number of inception-resnet units,  $\dim(x_i^C)$ : Resolution and dimensions of the output feature. Refer to [24, 26, 46, 47] for details

VGG16				ResNet101				InceptionResNetv2					
name	$C_i$	N	$\dim(x_i^C)$	name	$C_i$	NB	N	$\dim(x_i^C)$	name	$C_i$	NI	N	$\dim(x_i^C)$
conv1_x	$C_1$	2	(300, 500, 64)	conv1	$C_1$	1	1	(300, 500, 64)	conv_x	$C_x$	-	5	(71, 246, 192)
conv2_x	$C_2$	2	(150, 250, 128)	conv2_x	$C_2$	3	9	(150, 250, 256)	Mixed_5b	$M_5$	1	7	(35, 122, 320)
conv3_x	$C_3$	3	(75, 125, 256)	conv3_x	$C_3$	4	12	(75, 125, 512)	Block_10x	$B_{10}$	10	70	(35, 122, 320)
conv4_x	$C_4$	3	(37, 63, 512)	conv4_x	$C_4$	23	69	(75, 125, 1024)	Mixed_6a	$M_{6a}$	1	3	(33, 120, 1088)
conv5_x	$C_5$	3	(37, 63, 512)						Block_20x	$B_{20}$	20	100	(33, 120, 1088)

Due to lack of support for recent ConvNet architectures [24, 47] in the Faster R-CNN framework, we use our implementation in Tensorflow [1]. We follow the design choices outlined in [26]. In Section 5, we will provide performance numbers using both the released code [41] as well as our implementation (which tends to generally perform better). We use the end-to-end training paradigm for Faster R-CNN for all experiments [41]. Unless specified otherwise, all methods start with models that were pre-trained on ImageNet classification [8, 17].

## 4.2. Preliminaries: Base Network Architectures

In this paper, we use three standard ConvNet architectures: VGG16 [46], ResNet101 [24] and InceptionResNetv2 [47]. We briefly explain how they are incorporated in the Faster R-CNN framework (see [24, 26, 41] for details), and give a quick overview of these architectures with reference to the bottom-up blocks  $C$  from Section 3.1.

We use the term ‘Base Network’ to refer to the part of ConvNet that is shared by both RPN and RCN; and ‘Classifier Network’ to refer to the part that is used as RCN. For VGG16 [41, 46], ConvNet till `conv5_3` is used as the base network, and the following two `fc` layers are used as the classifier network. Similarly, for ResNet101 [24, 41], base network is the ConvNet till `conv4_x`, and classifier network is the `conv5_x` block (with 3 residual units or 9 convolutional layers). For InceptionResNet101v2 [26, 47], ConvNet till the ‘Block\_20x’ is used as the base network, and the remaining layers (‘Mixed\_7a’ and ‘Block\_9x’, with a total of 11 inception-resnet units or 48 convolutional layers) are used as the classifier network. Following [26], we change the pooling stride of the penultimate convolutional block in ResNet101 and InceptionResNetv2 to 1 to maintain spatial resolution, and use atrous [6, 34] convolution to recover the original field-of-view. Properties of bottom-up blocks  $C_i$ , including number of layers, the output feature resolution and feature dimension *etc.* are given in Table 1.

## 4.3. Top-Down Modulation

To add the proposed TDM network to the ConvNet architectures described above, we need to decide the extent of top-down modulation, the frequency of lateral connections

Table 2. **Top-Down Modulation** network design for VGG16, ResNet101 and InceptionResNetv2. Notice that  $t_{\text{out}}$  VGG16 is much smaller than 512, thus requiring fewer parameters in RPN and RCN modules. Also note that it is important to keep  $t_{\text{out}}$  fixed for ResNet101 and InceptionResNetv2 in order to utilize pre-trained RPN and RCN modules

VGG16					ResNet101				
$T_{i,j}$	$L_j$	$t_{i,j}$	$l_j$	$t_{\text{out}}$	$T_{i,j}$	$L_j$	$t_{i,j}$	$l_j$	$t_{\text{out}}$
$T_{5,4}$	$L_4$	128	128	256	$T_{4,3}$	$L_3$	128	128	1024
$T_{4,3}$	$L_3$	64	64	128	$T_{3,2}$	$L_2$	128	128	1024
$T_{3,2}$	$L_2$	64	64	128	$T_{2,1}$	$L_1$	32	32	1024
$T_{2,1}$	$L_1$	64	64	128					

InceptionResNetv2				
$T_{i,j}$	$L_j$	$t_{i,j}$	$l_j$	$t_{\text{out}}$
$T_{B_{20},6a}$	$L_{6a}$	576	512	1088
$T_{6a,5b}$	$L_{5b}$	512	256	1088

and the capacity of  $T$ ,  $L$  and  $T^{\text{out}}$  modules. We try to follow these principles when making design decisions: (a) coarse semantic modules need larger capacity; (b) lateral and top-down connections should reduce feature dimensionality in order to force selection; and (c) the capacity of  $T^{\text{out}}$  should be informed by the Proposal (RPN) and Classifier (RCN) Network design. Finally, the hardware constraint that a TDM augmented ConvNet should fit on a standard GPU.

To build a TDM network, we start with a standard bottom-up model trained on the detection task, and add  $(T_{i,j}, L_j)$  progressively. The capacity for different  $T$ ,  $L$ , and  $T^{\text{out}}$  modules is given in Table 2. For the VGG16 network, we add top-down and lateral modules all the way to the `conv1_x` feature. Notice that the input feature dimension to RPN and RCN networks changes from 512 (for `conv5_x`) to 256 (for  $T_4^{\text{out}}$ ), therefore we initialize the `fc` layers in RPN and RCN randomly [19]. However, since  $t_{\text{out}}$  is same for the last three  $T^{\text{out}}$  modules, we re-use the RPN and RCN layers for these modules.

For the ResNet101 and InceptionResNetv2, we add top-down and lateral modules for till `conv1` and ‘Mixed\_5b’ respectively. Similar to VGG16, their base networks are initialized with a model pre-trained on the detection task. However, as opposed to VGG16, where the RCN has just 2

Table 3. Object detection results on the COCO benchmark. Different methods use different networks for region proposal generation (ROI<sup>Net</sup>) and for region classification (Cls<sup>Net</sup>). Results for the top block (except Faster R-CNN\*) were directly obtained from their respective publications [24, 39, 41, 44]. Faster R-CNN\* was reproduced by us. Middle block shows our implementation of Faster R-CNN framework, which we use as our primary baseline. Bottom block presents the main results of TDM network, with current state-of-the-art single-model performance highlighted.

Method	train	test	ROI <sup>Net</sup>	Cl <sup>s</sup> Net	AP	AP <sup>50</sup>	AP <sup>75</sup>	AP <sup>S</sup>	AP <sup>M</sup>	AP <sup>L</sup>	AR <sup>1</sup>	AR <sup>10</sup>	AR <sup>100</sup>	AR <sup>S</sup>	AR <sup>M</sup>	AR <sup>L</sup>
Faster R-CNN [41]	train	val	VGG16	VGG16	21.2	41.5	-	-	-	-	-	-	-	-	-	-
Faster R-CNN [24]	train	val	ResNet101	ResNet101	27.2	48.4	-	-	-	-	-	-	-	-	-	-
SharpMask [39]	train	testdev	ResNet50	VGG16	25.2	43.4	-	-	-	-	-	-	-	-	-	-
Faster R-CNN [41]*	trainval	testdev	VGG16	VGG16	24.5	46.0	23.7	8.2	26.4	36.9	24.0	34.8	35.5	13.4	39.2	54.3
[44]	trainval	testdev	VGG16++	VGG16++	27.5	49.2	27.8	8.9	29.5	41.5	25.5	37.4	38.3	14.6	42.5	57.4
Faster R-CNN	trainval*	testdev	VGG16	VGG16	23.3	44.7	21.5	9.4	27.1	32.0	22.7	36.8	39.4	18.3	44.0	56.2
Faster R-CNN	trainval*	testdev	ResNet101	ResNet101	31.5	52.8	33.3	13.6	35.4	44.5	28.0	43.6	45.8	22.7	51.2	64.1
Faster R-CNN	trainval*	testdev	IRNv2	IRNv2	34.7	55.5	36.7	13.5	38.1	52.0	29.8	46.2	48.9	23.2	54.3	70.8
TDM [ours]	trainval*	testdev	VGG16 + TDM	VGG16 + TDM	28.6	48.1	30.4	14.2	31.8	36.9	26.2	42.2	44.2	23.7	48.3	59.3
TDM [ours]	trainval*	testdev	ResNet101 + TDM	ResNet101 + TDM	35.2	55.3	38.1	16.6	38.4	47.9	30.4	47.8	50.3	27.8	54.9	67.6
TDM [ours]	trainval*	testdev	IRNv2 + TDM	IRNv2 + TDM	<b>37.3</b>	<b>57.8</b>	<b>39.8</b>	<b>17.1</b>	<b>40.3</b>	<b>52.1</b>	<b>31.6</b>	<b>49.3</b>	<b>51.9</b>	<b>28.1</b>	<b>56.6</b>	<b>71.1</b>

fc layers, ResNet101 and InceptionResNetv2 models have an RCN with 9 and 48 convolutional layers respectively. This makes training RCN from random initialization difficult. To counter this, we ensure that all  $\mathbf{T}^{\text{out}}$  output feature dimensions ( $t_{\text{out}}$ ) are same, so that we can be readily use pre-trained RPN and RCN. This is implemented using an additional  $1 \times 1$  convolutional layer wherever ( $t_{i,j} + l_j$ ) differs from  $t_{\text{out}}$  (e.g., all  $\mathbf{T}^{\text{out}}$  modules in ResNet101, and the final  $\mathbf{T}^{\text{out}}$  module in InceptionResNetv2).

We would like to highlight an issue with training of RPN at high-resolution feature maps. RPN is a fully convolutional module of Faster R-CNN, that generates an intermediate 512 dimension representation which is of the same resolution as input; and losses are computed at all pixel locations. This is efficient for coarse features (e.g., last row in Table 1), but the training becomes prohibitively slow for finer resolution features. To counter this, we apply RPN at a stride which ensures that computation remains exactly the same (e.g., using stride of 8 for  $\mathbf{T}_1^{\text{out}}$  in VGG16). Because of ‘ROI Pooling’ operations, RCN module still efficiently utilizes the finer resolution features.

## 5. Results

In this section, we evaluate our method on the task of object detection, and demonstrate consistent and significant improvement in performance when using features from the proposed TDM network.

**Dataset and metrics.** All experiments and analysis in this paper are performed on the COCO dataset [33]. All models were trained on 40k train and 32k val images (which we refer to as ‘trainval\*’ set). All ablation evaluations were performed on 8k val images (‘minival\*’ set) held out from the val set. We also report quantitative results on the standard testdev2015 split. For quantitative evaluation, we use the

COCO evaluation metric of mean average precision (AP<sup>1</sup>).

**Experimental Setup.** We conduct experiments with three standard ConvNet architectures: VGG16 [46], ResNet101 [24] and InceptionResNetv2 [47]. All models (‘Baseline’ Faster R-CNN and ours) were trained with SGD for 1.5M mini-batch iterations, with batch size of 256 and 128 ROIs for RPN and RCN respectively. We start with an initial learning rate of 0.001 and decay it by 0.1 at 800k and 900k iterations.

**Baselines.** Our primary baseline is using vanilla VGG16, ResNet101 and InceptionResNetv2 features in the Faster R-CNN framework. However, due to lack of implementations supporting all three ConvNets, we opted to re-implement Faster R-CNN in Tensorflow [1]. The baseline numbers reported in Table 3(middle) are using our implementation and training schedule and are generally higher than the ones reported in [41, 44]. Faster R-CNN\* was reproduced by us using the official implementation [41]. All other results in Table 3(top) were obtained from the original papers.

We also compare against models which use a single region proposal generator and a single region classifier network. In particular, we compare with SharpMask [39], because of its refinement modules with top-down and lateral connections, and [44] because they also augment the standard VGG16 network with top-down information. Note that different methods use different networks and train/test splits (see Table 3), making it difficult to do a comprehensive comparison. Therefore, for discussion, we will directly compare against our Faster R-CNN baseline (Table 3(middle)), and highlight that the improvements obtained by our approach are much bigger than the boosts by other methods.

<sup>1</sup>COCO [33] AP averages over classes, recall, and IoU levels.

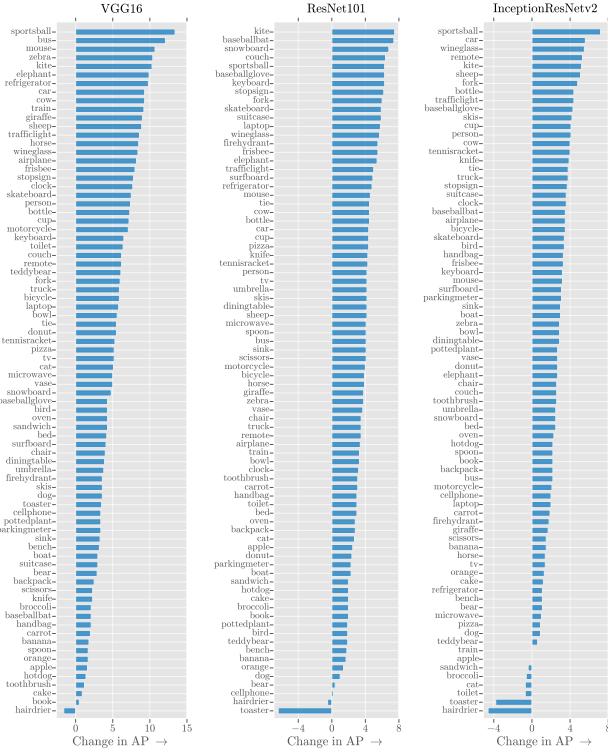


Figure 5. Improvement in AP over Faster R-CNN baseline. Base Networks: (left) VGG16, (middle) ResNet101, and (right) InceptionResNetv2. Improved performance for almost all categories emphasize the effectiveness of Top-Down Modulation for object detection. (best viewed digitally)

## 5.1. COCO Results

In Table 3(bottom), we report results of the proposed TDM network on the testdev2015 split of the COCO dataset. We see that the TDM network leads to a **5.3** AP point boost over the vanilla VGG16 network (28.6 AP vs. 23.3 AP), indicating that TDM learns much better features for object detection. Note that even though our algorithm is trained on less data (trainval\*), we outperform all methods with VGG16 architecture. For ResNet101, we improve the performance by **3.7** points to 35.2 AP. InceptionResNetv2 [47] architecture was the cornerstone of the winning entry to COCO 2016 detection challenge [26]. The best single model performance used by this entry achieves 34.7 AP on the testdev split ([26]). Using InceptionResNetv2 as base, our TDM network achieves **37.3** AP, which is currently the **state-of-the-art** single model performance on the testdev split without any bells and whistles (*e.g.*, multi-scale, iterative box refinement, *etc.*). In fact, the TDM network outperforms the baselines (with same base networks) on almost all AP and AR metrics. Similarly, in Table 4, we observe that TDM achieves similar boosts across all network architectures on the minival\* split as well.

Figure 5 shows change in AP from Faster R-CNN base-

line to the TDM network (for the testdev split). When using VGG16, for all but one category, TDM features improve the performance on object detection. In fact, more than 50% of the categories improve by 5 AP points or more, highlighting that the features are good for small and big objects alike. Similar trends hold for ResNet101 and InceptionResNetv2.

**Improved localization.** In Table 3, we also notice that for VGG16, our method performs exceptionally well on the AP<sup>75</sup> metric, improving the baseline Faster R-CNN by **8.9** AP<sup>75</sup> points, which is much higher than the **3.5** point AP<sup>50</sup> boost. We believe that using contextual features to select and integrate low-level finer details is they key reason for this improvement. Similarly for ResNet101 and InceptionResNetv2, we see **4.8** AP<sup>75</sup> and **3.1** AP<sup>75</sup> boost respectively.

**Improvement for small objects.** In Table 3, for VGG16, ResNet101 and InceptionResNetv2), we see **4.8**, **3** and **3.6** point boost respectively for small objects (AP<sup>S</sup>) highlighting the effectiveness of features with TDM. Moreover, small objects are often on top of the list in Figure 5 (*e.g.*, sportsball +13 AP point, mouse +10 AP for VGG16). This is in line with other studies [20], which show that context is particularly helpful for some objects. Similar trends hold for the minival\* split as well: **5.6**, **7.4** and **8.5** AP<sup>S</sup> boost for VGG16, ResNet101 and InceptionResNetv2 respectively.

**Qualitative Results.** In Figure 6, we display qualitative results of the Top-down Modulation Network. Notice that the network can find small objects like remote (second row, last column; fourth row, first column) and sportsball (second row, third and fourth column). Also notice the detection results in the presence of heavy clutter.

## 6. Design and Ablation Analysis

In this section, we perform control and ablative experiments to study the importance of top-down and lateral modules in the proposed TDM network.

### 6.1. How low should the Top-Down Modulation go?

In Section 4.3, we discussed the principles that we follow to add top-down and lateral modules. We connected these modules till the lowest layer choosing design decisions that hardware constraints would permit. However, is that overkill? Is there an optimal layer, after which this modulation does not help or starts hurting? To answer these questions, we limit the layer till which the modulation process happens, and use those features for Faster R-CNN.

Recall that the TDN network is built progressively, *i.e.*, we add one pair of lateral and top-down module at a time. So for this control study, we simply let each subsequent pair train for the entire learning schedule, treating the  $T^{out}$  as the final output. We report the results on minival\* set in Table 4.

Table 4. Ablation analysis on the COCO benchmark using the Faster R-CNN detection framework. All methods are trained on trainval\* and evaluated on minival\* set (Section 5). Methods are grouped based on their base network, **best** results are highlighted in each group.

Method	Net	Features from:	AP	AP <sup>50</sup>	AP <sup>75</sup>	AP <sup>S</sup>	AP <sup>M</sup>	AP <sup>L</sup>	AR <sup>1</sup>	AR <sup>10</sup>	AR <sup>100</sup>	AR <sup>S</sup>	AR <sup>M</sup>	AR <sup>L</sup>
Baseline	VGG16	C <sub>5</sub>	25.5	46.7	24.6	6.1	23.3	37.0	23.9	38.2	40.7	14.1	39.5	55.3
Skip-pool	VGG16	C <sub>2</sub> , C <sub>3</sub> , C <sub>4</sub> , C <sub>5</sub>	25.3	46.3	25.9	9.1	24.0	36.0	24.6	40.0	42.4	18.6	41.8	54.1
TDM [ours]	VGG16 + TDM	T <sub>4</sub> <sup>out</sup>	26.2	45.7	27.2	9.4	25.1	34.8	25.0	40.7	43.0	18.7	43.1	54.7
TDM [ours]	VGG16 + TDM	T <sub>3</sub> <sup>out</sup>	28.8	48.6	30.7	11.0	27.1	37.3	26.5	42.7	45.0	21.1	44.2	56.4
TDM [ours]	VGG16 + TDM	T <sub>2</sub> <sup>out</sup>	<b>29.9</b>	<b>50.3</b>	31.6	11.4	<b>28.1</b>	38.6	<b>27.3</b>	<b>43.7</b>	<b>46.0</b>	22.8	44.7	<b>57.1</b>
TDM [ours]	VGG16 + TDM	T <sub>1</sub> <sup>out</sup>	29.8	49.9	<b>31.7</b>	<b>11.7</b>	28.0	<b>39.3</b>	27.1	43.5	45.9	<b>23.9</b>	<b>45.4</b>	56.8
Baseline	ResNet101	C <sub>5</sub>	32.1	53.2	33.8	9.4	29.7	45.7	28.3	44.3	46.7	19.3	46.3	60.9
TDM [ours]	ResNet101 + TDM	T <sub>3</sub> <sup>out</sup>	34.4	54.4	37.1	10.9	31.8	48.2	30.1	47.5	49.8	21.7	49.1	64.0
TDM [ours]	ResNet101 + TDM	T <sub>2</sub> <sup>out</sup>	35.3	55.1	38.3	11.2	33.0	48.2	30.7	48.0	50.5	22.5	50.1	63.6
TDM [ours]	ResNet101 + TDM	T <sub>1</sub> <sup>out</sup>	<b>35.7</b>	<b>56.0</b>	<b>38.5</b>	<b>16.8</b>	<b>39.2</b>	<b>49.0</b>	<b>30.9</b>	<b>48.5</b>	<b>50.9</b>	<b>28.1</b>	<b>55.6</b>	<b>68.5</b>
Baseline	IRNv2	B <sub>20</sub>	35.7	56.5	38.0	8.9	32.0	52.5	30.8	47.8	50.3	19.6	49.9	66.9
TDM [ours]	IRNv2 + TDM	T <sub>6a</sub> <sup>out</sup>	37.3	57.9	39.5	11.4	33.3	53.3	<b>32.8</b>	49.1	51.5	22.7	50.6	67.5
TDM [ours]	IRNv2 + TDM	T <sub>5a</sub> <sup>out</sup>	<b>38.1</b>	<b>58.6</b>	<b>40.7</b>	<b>17.4</b>	<b>41.1</b>	<b>54.7</b>	32.4	<b>50.1</b>	<b>52.6</b>	<b>28.9</b>	<b>57.2</b>	<b>72.3</b>

Table 5. (left) **Importance of lateral modules**: We use operations from the top-down module to increase depth of the VGG16 network;  $\sim \mathbf{T}_{i,j}$  represents modified top-down module to account for more parameters. (right) **Impact of Pre-training**.

$i, j$	$\sim \mathbf{T}_{i,j}$	$(\mathbf{T}_{i,j}, \mathbf{L}_j)$	Pre-trained on:	
			COCO	ImageNet
5, 4	24.8	26.2		
4, 3	25.1	28.8	$\mathbf{T}_{4,3}$	34.4
3, 2	26.5	<b>29.9</b>	$\mathbf{T}_{3,2}$	34.1
2, 1	21.4	29.6		

As we can see, adding more top-down modulation helps in general. However, for VGG16, we see that the performance saturates at  $\mathbf{T}_2^{\text{out}}$ , and adding modules till  $\mathbf{T}_1^{\text{out}}$  do not seem to help much. Deciding the endpoint criteria for top-down modulation is an interesting future direction.

## 6.2. No lateral modules

To analyze the importance of lateral modules, and to control for the extra parameters added by the TDM network (Table 2), we train additional baselines with variants of VGG16 + TDM network. In particular, we remove the lateral modules and use convolutional and upsampling operations from the top-down modules  $\mathbf{T}$  to train ‘deeper’ variants of VGG16 as baseline. To control for the extra parameters from lateral modules, we also increase the parameters in the convolutional layers. Note that for this baseline, we follow training methodology and design decisions used for training TDM networks.

As shown in Table 5(left), even though using more depth increases the performance slightly, the performance boost due to lateral modules is much higher. This highlights the importance of dividing the capacity of TDM network amongst lateral and top-down modules.

## 6.3. No top-down modules

Next we want to study the importance of the top-down path introduced by our TDM network. We believe that this path is responsible for transmitting contextual features and for selection of relevant finer details. Removing the top-down path exposes the ‘skip’-connections from bottom-up features, which can be used for object detection. We follow the strategy from [4], where they ROI-pool features from different layers, L2-normalize and concatenate these features and finally scale them back to the original conv5\_3 magnitude and dimension.

We tried many variants of the Skip-pooling baseline, and report the best results in Table 4 (Skip-pool). We see that performance for small objects (AP<sup>S</sup>) increases slightly, but overall the AP does not change much. This highlights the importance of using high-level contextual features in the top-down path for the selection of low-level features.

## 6.4. Impact of Pre-training

Finally, we study the impact of using a model pre-trained on the detection task to initialize our base networks and ResNet101/InceptionResNetv2’s RPN and RCN networks vs. using only an image classification [8] pre-trained model. In Table 5(right), we see that initialization does not impact the performance by a huge margin. However, pre-training on the detection task is consistently better than using the classification initialization.

## 7. Conclusion

This paper introduces the Top-Down Modulation (TDM) network, which leverages top-down contextual features and lateral connections to bottom-up features for object detection. The TDM network uses top-down context to select low-level finer details, and learns to integrate them together. Through extensive experiments on the challenging COCO

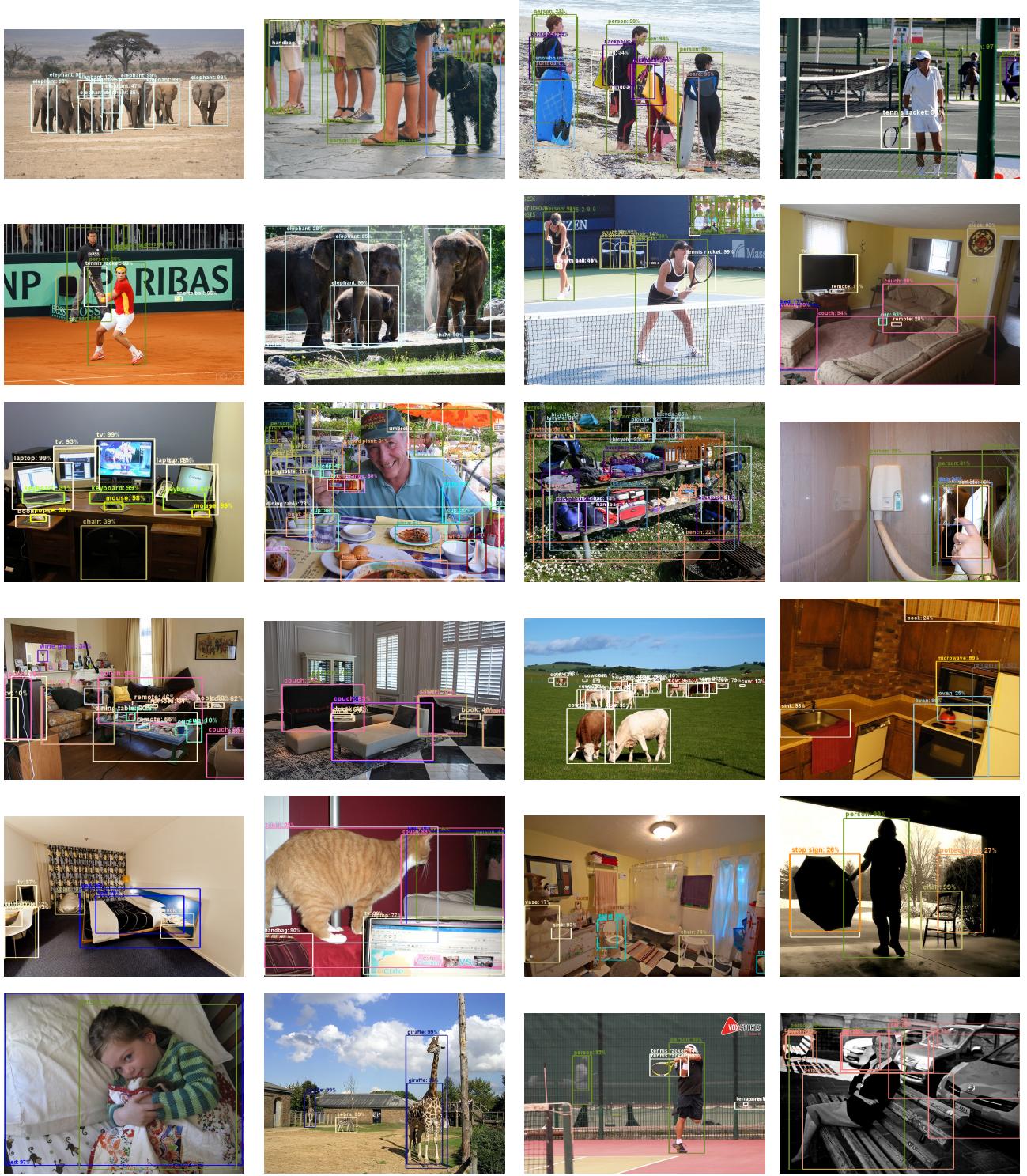


Figure 6. **Qualitative results** of the proposed TDM network on randomly selected images from the minival\* set (best viewed digitally).

dataset, we demonstrate the effectiveness and importance of features from the TDM network. We show empirically that the proposed representation benefits all objects, big and small, and is helpful for accurate localization.

Even though we focused on the object detection, we believe these top-down modulated features will be helpful in a wide variety of computer vision tasks.

## References

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- [2] T. Bachmann. A hidden ambiguity of the term feedback in its use as an explanatory mechanism for psychophysical visual phenomena. *Feedforward and Feedback Processes in Vision*, 2015.
- [3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.
- [4] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *IEEE CVPR*, 2015.
- [5] J. Carreira, P. Agrawal, K. Fragiadaki, and J. Malik. Human pose estimation with iterative error feedback. *arXiv preprint arXiv:1507.06550*, 2015.
- [6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.
- [7] M. M. Chun and Y. Jiang. Top-down attentional guidance based on implicit learning of visual covariation. *Psychological Science*, 1999.
- [8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR*, 2009.
- [9] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *IEEE ICCV*, 2015.
- [10] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE TPAMI*, 2013.
- [11] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.
- [12] C. Gatta, A. Romero, and J. van de Veijer. Unrolling loopy top-down semantic feedback in convolutional deep networks. In *IEEE CVPR Workshops*, 2014.
- [13] A. Gazzaley and A. C. Nobre. Top-down modulation: bridging selective attention and working memory. *Trends in cognitive sciences*, 2012.
- [14] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *IEEE ICCV*, 2015.
- [15] C. D. Gilbert and M. Sigman. Brain states: top-down influences in sensory processing. *Neuron*, 2007.
- [16] R. Girshick. Fast R-CNN. In *ICCV*, 2015.
- [17] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE CVPR*, 2014.
- [18] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with RCNN. In *IEEE ICCV*, 2015.
- [19] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.
- [20] S. Gupta, B. Hariharan, and J. Malik. Exploring person context and local scene context for object detection. *arXiv preprint arXiv:1511.08177*, 2015.
- [21] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.
- [22] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *IEEE CVPR*, 2015.
- [23] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.
- [24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
- [25] J. B. Hopfinger, M. H. Buonocore, and G. R. Mangun. The neural mechanisms of top-down attentional control. *Nature neuroscience*, 2000.
- [26] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017.
- [27] D. J. Kravitz, K. S. Saleem, C. I. Baker, L. G. Ungerleider, and M. Mishkin. The ventral visual pathway: an expanded neural framework for the processing of object quality. *Trends in cognitive sciences*, 2013.
- [28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
- [29] V. A. Lamme and P. R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in neurosciences*, 2000.
- [30] V. A. Lamme, H. Super, and H. Spekreijse. Feedforward, horizontal, and feedback processing in the visual cortex. *Current opinion in neurobiology*, 1998.
- [31] K. Li, B. Hariharan, and J. Malik. Iterative instance segmentation. *arXiv preprint arXiv:1511.08498*, 2015.
- [32] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*, 2016.
- [33] 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 *ECCV*, 2014.
- [34] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.
- [35] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, 2015.
- [36] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE ICCV*, 2015.
- [37] V. Piëch, W. Li, G. N. Reeke, and C. D. Gilbert. Network model of top-down influences on local gain and contextual interactions in visual cortex. *Proceedings of the National Academy of Sciences*, 2013.
- [38] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *NIPS*, 2015.
- [39] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. *arXiv preprint arXiv:1603.08695*, 2016.
- [40] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *arXiv preprint arXiv:1611.00850*, 2016.
- [41] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal net-

- works. In *NIPS*, 2015.
- [42] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.
  - [43] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *IEEE CVPR*, 2013.
  - [44] A. Shrivastava and A. Gupta. Contextual priming and feedback for Faster R-CNN. In *ECCV*, 2016.
  - [45] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *IEEE CVPR*, 2016.
  - [46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
  - [47] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.
  - [48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE CVPR*, 2015.
  - [49] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, 2013.
  - [50] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014.
  - [51] A. Toshev and C. Szegedy. Deppose: Human pose estimation via deep neural networks. In *IEEE CVPR*, 2014.
  - [52] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *IEEE CVPR*, 2015.
  - [53] S. Xie and Z. Tu. Holistically-nested edge detection. In *IEEE ICCV*, 2015.
  - [54] T. P. Zanto, M. T. Rubens, J. Bollinger, and A. Gazzaley. Top-down modulation of visual feature processing: the role of the inferior frontal junction. *Neuroimage*, 2010.
  - [55] T. P. Zanto, M. T. Rubens, A. Thangavel, and A. Gazzaley. Causal role of the prefrontal cortex in top-down modulation of visual processing and working memory. *Nature neuroscience*, 2011.
  - [56] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *IEEE CVPR*, 2010.