SINGLE SHOT OBJECT DETECTION WITH TOP-DOWN REFINEMENT

Guangxing Han, Xuan Zhang, Chongrong Li

Institute for Network Sciences and Cyberspace (INSC),
Tsinghua National Laboratory for Information Science and Technology (TNList),
Tsinghua University, Beijing, 100084, China.
hgx14@mails.tsinghua.edu.cn, {zhangx, licr}@cernet.edu.cn

ABSTRACT

General object detection is one of the most challenging tasks in computer vision for it requires both high running speed and detection accuracy. In this paper, we propose a single shot object detector with top-down refinement, denoted as SSD-TDR. It not only runs at high speed and also detects multi-scale objects accurately. Concretely, original SSD directly adopts the built-in multi-scale hierarchy of convolutional neural networks for detection. However, object detection needs high semantic knowledge to recognise objects while low-level convolutional features do not have. We thus build a sequence of top-down refinement modules to transmit semantic knowledge backward such that all layers have rich semantics. Experiments on PASCAL VOC 2007 and 2012 demonstrate that our network achieves competitive results both in speed and accuracy compared to other VGG16 based networks.

Index Terms— convolutional neural network, general object detection, single shot detector, top-down refinement, multi-scale detection

1. INTRODUCTION

Recently computer vision community has been revolutionized by deep convolutional neural networks (DCNNs) [1]. In general object detection, tremendous progress has been made due to the use of DCNNs. State-of-the-art object detection networks[2] such as Faster R-CNN [3], R-FCN [4] and SSD [5] achieve both high detection accuracy and running speed in several object detection benchmarks.

In Faster R-CNN and R-FCN, object detection is divided into two stages: Region Proposal Network (RPN) and RoI-wise Classification Network (RCN). Proposals are first generated through a RPN network and then those candidates are further classified and refined by a RCN network, which is too time-consuming. R-FCN shares per-region computation in RCN using position-sensitive pooling and achieves comparable accuracy to Faster R-CNN with faster speed. But it still cannot meet the requirement of real-time detection. While SSD, evolved from Faster R-CNN and YOLO [6, 7], employing a single feed-forward convolutional network to directly predict bounding boxes and their classes, thus leading to faster running speed than the previous two models. SSD still maintains high detection accuracy by simultaneously detecting multi-scale objects from multiple feature layers. Moreover, a variant of networks derived from SSD also achieve state-of-the-art results in many other computer vision problems beyond object detection, for example, [8] uses SSD-style network to estimate the 3D pose of objects and Textboxes [9] adopts SSD architecture for text localization.

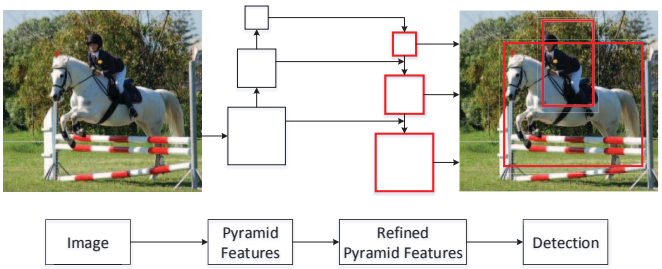

Fig. 1: Overview of our single shot object detector. Firstly we forward convolutional neural networks to build pyramidal multi-scale features. Then top-down refinement modules are adopted to enrich semantic infomation at all scales. Finally, objects are detected at all layers simultaneously.

However, there are still problems for accurate localization in SSD especially for small objects. Concretely, these three detection networks including SSD are all fine-tuned from modern DCNNs such as VGGNet [10] and ResNet [11]. They are all bottom-up, feed-forward architectures and use repeated convolutional layers and pooling layers to learn strong feature representations for the input image. In DCNNs, higher layers are usually more effective to capture high-level semantic knowledge, but insufficient for capturing fine-grained spatial details due to its low resolution through repeated pooling layers. Small objects usually have very small features maps on higher layers which is too coarse for localization. While lower layers feature maps have relatively large resolution but less semantic knowledge which is not accurate enough for small object detection.

To solve these problems, [12, 13, 14] proves that seman-

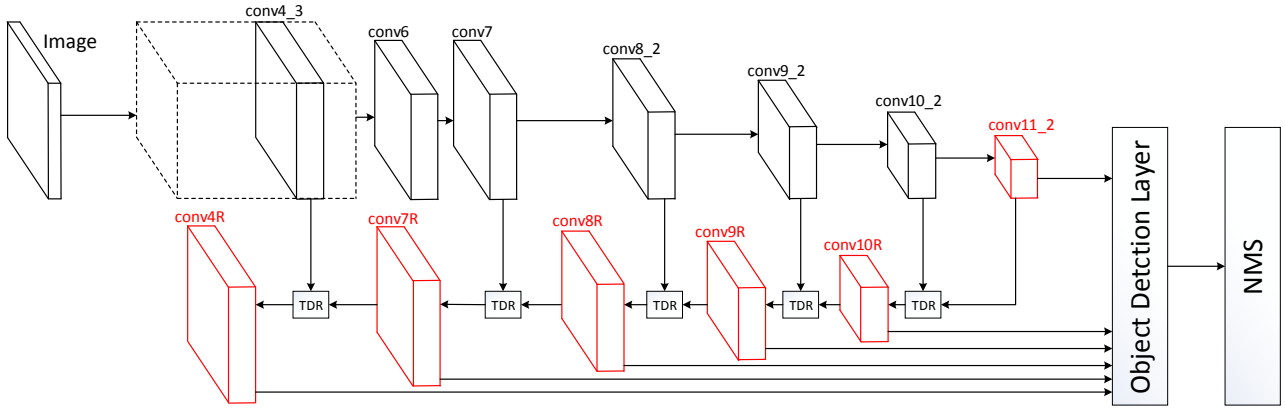

Fig. 2: The architecture of our SSD-TDR. Our model is based on VGG16. On top of conv5_3, several feature layers are added to build multi-scale feature maps. Moreover, TDR modules are adopted to refine these feature maps. Originally in SSD, multi-scale objects are predicted on layers conv4_3, conv7, conv8_2, conv9_2, conv10_2 and conv11_2 independently. In SSD-TDR, feature maps from layer conv4R to conv10R together with conv11_2 are responsible for final multi-scale object detection.

tics and context information help a lot for detection including small scale objects. [15, 16, 17, 18] use skip connections to directly combine multi-layer features for final strong representations. Thus semantic knowledge and context information are fused into high-resolution lower layer feature maps. Another method is to use top-down refinement, such as FPN [19], TDM [20] and SharpMask [21]. Skip connections is a special case of this process [20]. In FPN and TDM, feature pyramids generated in forward pass are gradually supplemented with rich semantic knowledge at all scales by top-down refinement. After that two-stages Faster R-CNN are adopted for detection. SharpMask adopts similar top-down modulation to generate high-fidelity object masks.

In this paper, we propose a novel single shot object detector with both bottom-up and top-down passways. Firstly, initial multi-scale feature maps are built through a bottom-up feedforward. Then top-down refinement (TDR) modules are followed to enrich the semantic representation on each level of the feature pyramids. Finally, multi-scale objects are generated simultaneously on multiple feature layers. In addition, our network can be easily trained in a two-step manner. Experiments on PASCAL VOC 2007 and 2012 demonstrate that top-down refinement is effective to improve detection accuracy including small scale objects in our single shot network. Our main contribution is in two folds:

- We propose a novel top-down refinement to supplyment the standard single shot detector. Thus we predict bounding boxes on multi-scale feature maps all with high semantics and context knowledge.
- We achieves 78.3% and 76.2% mAP on VOC 2007 and 2012 respectively. Moreover, our network runs at 28 fps on TITAN X GPU due to our single shot architecture design. Both detection accuracy and running speed is competitive to other existing state-of-the-art VGG16 based object detection networks.

2. OUR APPROACH

2.1. Architecture

Whole pipeline. The detailed architecture of our detection network is shown in Figure 2. We use the *à trous* VGG16-net as our backbone network, keeping layers from conv1_1 to conv5_3 the same. Then change the stride of layer pool5 to 1, and change last two fully connection layers into dilated convolutional layers and subsampling the parameters as in [5]. On top of layer conv7, we build a few extra convolutional and pooling layers from conv8_1 to conv11_2 which progressively decrease spatial resolution. Then we add a sequence of top-down refinement modules from conv11_2 to conv4_3 to transmit semantic knowledge and context information gradually to lower layers. Thus all feature layers capture high-level infomation which is crucial for accurate object detection. After that, we predict multi-scale objects on all levels of our feature maps respectively. Finally we aggregate the predicted bounding boxes and use non maximum suppresion (NMS) to achieve final detection results.

Top-down refinement modules. Our top-down refinement module is shown in Figure 3. The inputs of our top-down module come from two parts. One is the feature maps generated in the feed-forward pass. The other is the refined features from the top-down way. Then these features are combined using our top-down refinement module. However, refined features from high-level usually have low-resolution, so deconvolutional layers are adopted to upsample features. Moveover, the corresponding bottom-up features usually have large feature channels, so we employ 1×1 convolution to reduce channel numbers such that it has the same channel di-

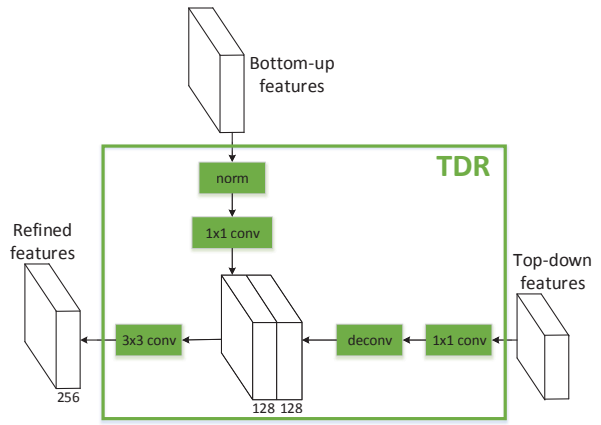

Fig. 3: Details of our TDR module. Bottom-up features and top-down features are combined through lateral connections.

mensions with top-down features. We also observe that lower layers, for example in VGGNet, usually has large responses, L2-normalization [22] is preferred before feature combination which leads to a faster converge and more stable training. After that, features from two sides are concatenated along the channel dimension and 3×3 convolution is followed to fuse these features sufficiently. In our network, final features at all levels are reduced to 256 channels. Similar modules are conducted from layer conv10R to conv4R. The architecture details are shown in table 1. Finally, we can get pyramid features all with high simantic knowledge.

SSD		SSD-TDR		Feature size
name	channel	name	channel	
conv4_3	512	conv4R	256	38x38
conv7	1024	conv7R	256	19x19
conv8_2	512	conv8R	256	10x10
conv9_2	256	conv9R	256	5x5
conv10_2	256	conv10R	256	3x3
conv11_2	256	conv11_2	256	1x1

Table 1: Architecture details of SSD and SSD-TDR. The input image is resized to 300x300. This table shows the layers responsible for detection in SSD and SSD-TDR.

Object detection layer. Object detection layers are another key components in our network. After getting refined pyramid features, we directly build 3×3 convolutional layers independently on each feature map to predict multi-scale bounding boxes. For example, in a final feature map with size $w \times h$, we predict a number of bounding boxes with multiple aspect ratios and their classification scores for each category at all of the $w \times h$ positions. Specifically, for bounding boxes, we output the offset value relative to the default box coordinates. Suppose that the default box $G = (G_x, G_y, G_w, G_h)$, our predicted box $P = (P_x, P_y, P_w, P_h)$, and our predicted offset values $(\Delta_x, \Delta_y, \Delta_w, \Delta_h)$. The relationships are:

$$\begin{aligned} P_x &= G_x + G_w * \Delta_x \quad P_y = G_y + G_h * \Delta_y \\ P_w &= G_w * \exp(\Delta_w) \quad P_h = G_h * \exp(\Delta_h) \end{aligned} \tag{1}$$

We apply this layer to several feature maps. Finally NMS is adopted to aggregate outputs of all detection layers.

2.2. Training

As there are two basic components bottom-up and top-down passways in our network, we introduce a two-step training mechanism to optimize our model. In the first step, we train our bottom-up network. We use different scales of default boxes on each feature layers as in [5] to fit their receptive fields. Then a default box is matched with a ground truth box p if it has the highest jaccard overlap with p or their jaccard overlap is higher than 0.5. The loss function is:

$$L(M, c, p, g) = \frac{1}{N}(L_{conf}(M, c) + \alpha L_{loc}(M, p, g)) \tag{2}$$

where **M** is the match matrix, c is the predicted confidence, p and q are predicted and ground-truth location, N is the number of matched default boxes. Non-matched boxes are ignored when calculating training loss. The confidence loss is the softmax loss over multiple classes. Smooth L1 loss [23] is adopted for localization loss. In the second step, we integrate top-down refinement modules into the trained bottom-up network. The loss function is the same as the first step. Newly added layers adopt "Xavier" initialization and are trained alone first, then whole network is jointly trained end-to-end.

2.3. Implemetation details

Our implementation is based on the publicly available code of SSD. The input image is resized to 300×300. Training images are augmented with random crop, color distortion and horizontal flip. We adopt synchronized SGD training on 2 GPUs, each with 16 images in a mini-batch. We use a weight decay of 0.0005 and a momentum of 0.9. In the first step, we adopt the same learning rate stategies as SSD.In the second step, we firstly train newly added layers alone. The learning rate is 10^{-3} for 20k mini-batches, 10^{-4} for next 10k. Then whole network is jointly trained. The learning rate is 10^{-3} for 20k mini-batches, 10^{-4} and 10^{-5} for next 20k and 10k mini-batches respectively. We also adopt hard negative mining to pick negative examples with large confidence loss such that the ratio of negatives and positives is at most 3:1.

2.4. Comparision with state-of-the-art networks

Concurrent with our work, TDM proposes top-down modulation to replace skip connections for transmitting semantic features backward. However, it generates proposals only on the final high resolution feature map. We argue that different

Approach	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster R-CNN	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
ION	**79.2**	**80.2**	**85.2**	**78.8**	70.9	**62.6**	86.6	**86.9**	89.8	**61.7**	**86.9**	76.5	88.4	87.5	83.4	**80.5**	52.4	78.1	77.2	86.9	**83.5**
YOLOv2-416	76.8	78.4	85.1	75.8	64.7	46.8	84.6	86.0	**89.9**	56.4	80.9	75.5	**88.9**	**88.2**	85.6	77.4	49.1	77.7	**80.7**	**88.1**	76.8
SSD-300†	77.7	79.2	84.0	75.7	70.0	50.9	**86.7**	86.0	88.6	60.1	81.4	76.8	86.3	87.3	84.2	79.5	52.7	79.4	79.4	87.7	77.2
SSD-TDR-300	78.3	80.0	85.1	78.1	**71.4**	50.2	**86.7**	86.4	88.6	**61.7**	82.9	**77.0**	86.3	87.3	**86.0**	79.8	**54.1**	**80.3**	77.9	**88.1**	77.3

Table 2: **PASCAL VOC 2007 test detection results.** All methods are based on VGG16 and trained with 2007trainval+2012trainval. Mean average precision and running speed are shown for a variety of methods. † Our own reproducing of SSD-300, slightly higher than [5].

Approach	FPS	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster R-CNN	7	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
ION	0.87	**76.4**	87.5	**84.7**	**76.8**	**63.8**	**58.3**	82.6	**79.0**	**90.9**	57.8	**82.0**	**64.7**	88.9	**86.5**	84.7	82.3	**51.4**	78.2	69.2	85.2	**73.5**
YOLOv2-544	40	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	**89.3**	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7
SSD-300	**46**	75.7	**88.0**	82.6	74.5	61.7	47.4	83.0	78.9	91.4	58.1	80.0	63.8	**89.3**	85.4	85.4	**82.5**	50.2	79.6	**73.5**	**86.6**	72.2
SSD-TDR-300	28	76.2†	87.1	83.5	75.3	62.3	48.4	**83.3**	**79.0**	90.8	**60.0**	81.5	64.5	88.9	86.4	**86.8**	82.4	50.6	**81.5**	73.3	86.2	72.1

Table 3: **PASCAL VOC 2012 test detection results.** All methods are based on VGG16 and trained with 2007trainvaltest+2012trainval. † `http://host.robots.ox.ac.uk:8080/anonymous/PDC8PS.html`

feature layers naturally have different receptive fields [24, 25] which is more suitable for certain scale object detection.

FPN is also based on two-stages Faster R-CNN framework. It exploits the multi-scale hierarchy of ConvNets with top-down architecture. In our model, we use slightly different lateral connections in the top-down passway for VGGNet compared to FPN. Moreover, as far as we know, our paper is the first to demonstrate that single shot detector benefits from top-down refined features and runs much faster than FPN.

3. EXPERIMENTAL RESULTS

Datasets. We comprehensively evaluate our method on PASCAL VOC detection benchmark [26]. There are 5011 trainval images, 4952 test images over 20 object categories in VOC 2007 and 11540 trainval images, 10991 test images over the same 20 categories in VOC 2012.

Evaluation Protocol. We adopt the standard mean average precision (mAP) evaluation for object detection. Detections are judged true/false based on the Intersection over Union (IoU) between predicted and ground-truth boxes. In PASCAL VOC, we evaluate mAP at IoU=0.5.

PASCAL VOC 2007 results. We evalute our network on PASCAL VOC 2007 test, trained on VOC 2007 trainval together with VOC 2012 trainval (16551 trainval images in total), which is the common pratice. The results are shown in table 2. The input image is resized to 300×300. Our network SSD-TDR-300 achieves 78.3% mAP, 0.6 points higher than original SSD300×300, and 1.5 points higher than YOLOv2 regardless of its higher input resolution 416×416. Specifically, detection on 14 classes out of the total 20 classes including small scale objects such as "plant" is improved when compared to original SSD300×300. This indicates that our top-down refinement helps a lot for accurate object detection among a wide variety of object scales. Moreover, our method also achieves significant higher mAP than those two-stages R-CNN based networks, 5.1 points higher than Faster R-CNN. We also notice that ION has 0.9 points higher mAP than us because it adds bells and whistles to its network, such as segmentation labels and iterative bounding-box regression.

model	model size (MB)	mAP
SSD-300	105.2	77.7
SSD-300 more convs	**113.1**	77.6
SSD-TDR-300	112.5	**78.3**

Table 4: Effects of the top down connections in our model on VOC 2007 test dataset.

PASCAL VOC 2012 results. We also evaluate on the slightly more challenging VOC 2012 test. The evalution result is shown in table 3. We can observe similar results with VOC 2007 test. Moreover, Our network runs about 28 fps on TITAN X GPU. This indicates that our method is the top detection method both in speed and accuracy compared to other VGG16 based networks.

Ablation experiments. Table 4 shows the ablation results of SSD-TDR without top down connections. Additional convolutions are retained to keep similar model size. However, these additional parameters benefit little to perfermance, which demonstrates the effectiveness of our TDR design.

4. CONCLUSIONS

In this paper, we propose a single shot object detection network with top-down refinement. Top-down refinement combines high-level semantic knowledge with low-level fine-grained features. Thus we build pyramid features all with rich sematic knowledge. Experiments on PASCAL VOC 2007 and 2012 demonstrate the effectiveness of our top-down refinement design. Moreover, we also obtain high running speed about 28fps for 300×300 images which is practicable for real-time applications.

5. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[2] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *arXiv:1611.10012*, 2016.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *PAMI*, 2016.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *NIPS*, 2016.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "SSD: single shot multibox detector," in *ECCV*, 2016.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[7] Joseph Redmon and Ali Farhadi, "Yolo9000: Better, faster, stronger," *arXiv:1612.08242*, 2016.

[8] Patrick Poirson, Phil Ammirato, Cheng-Yang Fu, Wei Liu, Jana Kosecka, and Alexander C. Berg, "Fast single shot detection and pose estimation," *arXiv:1609.05590*, 2016.

[9] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu, "Textboxes: A fast text detector with a single deep neural network," in *AAAI*, 2017.

[10] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[12] Zhaowei Cai, Quanfu Fan, Rogerio S. Feris, and Nuno Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *ECCV*, 2016.

[13] Peiyun Hu and Deva Ramanan, "Finding tiny faces," *arXiv:1612.04402*, 2016.

[14] Abhinav Shrivastava and Abhinav Gupta, "Contextual priming and feedback for faster r-cnn," in *ECCV*, 2016.

[15] Sean Bell, C. Lawrence Zitnick, Kavita Bala, and Ross Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *CVPR*, 2016.

[16] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun, "Hypernet: Towards accurate region proposal generation and joint object detection," in *CVPR*, 2016.

[17] Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro O. Pinheiro, Sam Gross, Soumith Chintala, and Piotr Dollár, "A multipath network for object detection," in *BMVC*, 2016.

[18] K. Kim, Y. Cheon, S. Hong, B. Roh, and M. Park, "PVANET: deep but lightweight neural networks for real-time object detection," *arXiv:1608.08021*, 2016.

[19] Tsung-Yi Lin, Piotr Dollr, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," *arXiv:1612.03144*, 2016.

[20] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta, "Beyond skip connections: Top-down modulation for object detection," *arXiv:1612.06851*, 2016.

[21] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár, "Learning to refine object segments," in *ECCV*, 2016.

[22] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," in *ICLR*, 2016.

[23] Ross Girshick, "Fast r-cnn," in *ICCV*, 2015.

[24] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *NIPS*, 2016.

[25] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba, "Object detectors emerge in deep scene cnns," in *ICLR*, 2015.

[26] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *IJCV*, 2015.