PVANET: Deep but Lightweight Neural Networks for Real-time Object Detection

Kye-Hyeon Kim, Yeongjae Cheon, Sanghoon Hong, Byungseok Roh, and Minje Park
Intel Imaging and Camera Technology
21 Teheran-ro 52-gil, Gangnam-gu, Seoul 06212, Korea
{kye-hyeon.kim, yeongjae.cheon, sanghoon.hong,
peter.roh, minje.park}@intel.com

Abstract

This paper presents how we can achieve the state-of-the-art accuracy in multi-category object detection task while minimizing the computational cost by adapting and combining recent technical innovations. Following the common pipeline of "CNN feature extraction + region proposal + RoI classification", we mainly redesign the feature extraction part, since region proposal part is not computationally expensive and classification part can be efficiently compressed with common techniques like truncated SVD. Our design principle is *"less channels with more layers"* and adoption of some building blocks including concatenated ReLU, Inception, and HyperNet. The designed network is deep and thin and trained with the help of batch normalization, residual connections, and learning rate scheduling based on plateau detection. We obtained solid results on well-known object detection benchmarks: 81.8% mAP (mean average precision) on VOC2007 and 82.5% mAP on VOC2012 (2nd place), while taking only 750ms/image on Intel i7-6700K CPU with a single core and 46ms/image on NVIDIA Titan X GPU. Theoretically, our network requires only 12.3% of the computational cost compared to ResNet-101, the winner on VOC2012.

1 Introduction

Convolutional neural networks (CNNs) have made impressive improvements in object detection for several years. Thanks to many innovative work, recent object detection systems have met acceptable accuracies for commercialization in a broad range of markets like automotive and surveillance. In terms of detection speed, however, even the best algorithms are still suffering from heavy computational cost. Although recent work on network compression and quantization shows promising result, it is important to reduce the computational cost in the network design stage.

This paper presents our lightweight feature extraction network architecture for object detection, named PVANET, which achieves real-time object detection performance without losing accuracy compared to the other state-of-the-art systems:

- Computational cost: 7.9GMAC for feature extraction with 1065x640 input (cf. ResNet-101 [1]: 80.5GMAC[1])
- Runtime performance: 750ms/image (1.3FPS) on Intel i7-6700K CPU with a single core; 46ms/image (21.7FPS) on NVIDIA Titan X GPU
- Accuracy: 81.8% mAP on VOC-2007; 82.5% mAP on VOC-2012 (2nd place)

[1]ResNet-101 used multi-scale testing without mentioning additional computation cost. If we take this into account, ours requires only <7% of the computational cost compared to ResNet-101.

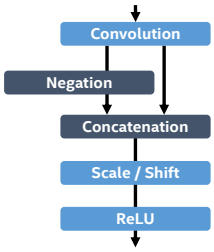

Figure 1: Our C.ReLU building block. **Negation** simply multiplies -1 to the output of Convolution. **Scale / Shift** applies trainable weight and bias to each channel, allowing activations in the negated part to be adaptive.

The key design principle is "less channels with more layers". Additionally, our networks adopted some recent building blocks while some of them have not been verified their effectiveness on object detection tasks:

- Concatenated rectified linear unit (C.ReLU) [2] is applied to the *early stage* of our CNNs (i.e., first several layers from the network input) to reduce the number of computations by half without losing accuracy.
- Inception [3] is applied to the remaining of our feature generation sub-network. An Inception module produces output activations of different sizes of receptive fields, so that increases the variety of receptive field sizes in the previous layer. We observed that stacking up Inception modules can capture widely varying-sized objects more effectively than a linear chain of convolutions.
- We adopted the idea of multi-scale representation like HyperNet [4] that combines several intermediate outputs so that multiple levels of details and non-linearities can be considered simultaneously.

We will show that our thin but deep network can be trained effectively with batch normalization [5], residual connections [1], and learning rate scheduling based on plateau detection [1].

In the remaining of the paper, we describe our network design briefly (Section 2) and summarize the detailed structure of PVANET (Section 3). Finally we provide some experimental results on VOC-2007 and VOC-2012 benchmarks, with detailed settings for training and testing (Section 4).

2 Details on Network Design

2.1 C.ReLU: Earlier building blocks in feature generation

C.ReLU is motivated from an interesting observation of intermediate activation patterns in CNNs. In the early stage, output nodes tend to be "paired" such that one node's activation is the opposite side of another's. From this observation, C.ReLU reduces the number of output channels by half, and doubles it by simply concatenating the same outputs *with negation*, which leads to 2x speed-up of the early stage without losing accuracy.

Figure 1 illustrates our C.ReLU implementation. Compared to the original C.ReLU, we append scaling and shifting after concatenation to allow that each channel's slope and activation threshold can be different from those of its opposite channel.

2.2 Inception: Remaining building blocks in feature generation

For object detection tasks, Inception has neither been widely applied to existing work, nor been verified its effectiveness. We found that Inception can be one of the most cost-effective building block for capturing both small and large objects in an input image. To Learn visual patterns for capturing large object, output features of CNNs should correspond to sufficiently large receptive fields, which can be easily fulfilled by stacking up convolutions of 3x3 or larger kernels. On the

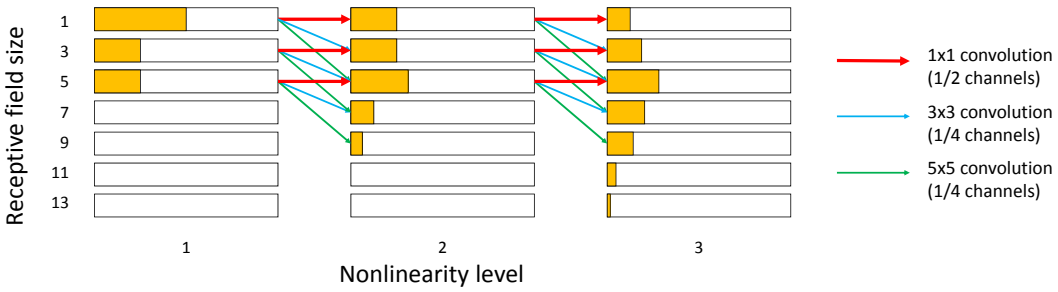

Figure 2: Example of a distribution of (expected) receptive field sizes of intermediate outputs in a chain of 3 Inception modules. Each module concatenates 3 convolutional layers of different kernel sizes, 1x1, 3x3 and 5x5, respectively. The number of output channels in each module is set to $\{1/2, 1/4, 1/4\}$ of the number of channels from the previous module, respectively. A latter Inception module can learn visual patterns of wider range of sizes, as well as having higher level of nonlinearity.

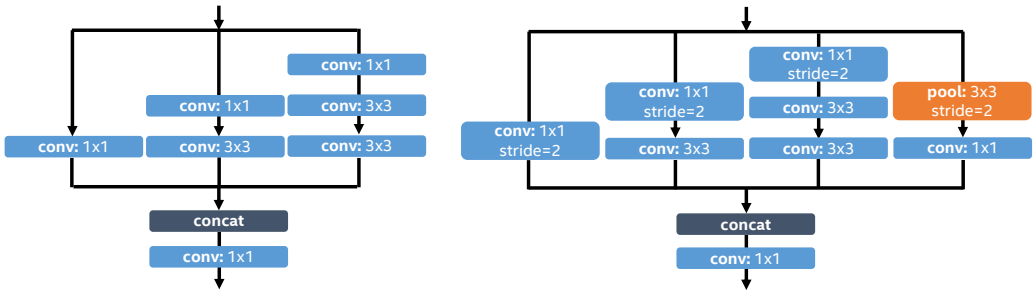

Figure 3: (Left) Our Inception building block. 5x5 convolution is replaced with two 3x3 convolutional layers for efficiency. (Right) Inception for reducing feature map size by half.

other hand, for capturing small-sized objects, output features should correspond to sufficiently small receptive fields to localize small regions of interest precisely.

Figure 2 clearly shows that Inception can fulfill both requirements. 1x1 convolution plays the key role to this end, by *preserving the receptive field* of the previous layer. Just increasing the nonlinearity of input patterns, it slows down the growth of receptive fields for some output features so that small-sized objects can be captured precisely. Figure 3 illustrates our Inception implementation. 5x5 convolution is replaced with a sequence of two 3x3 convolutions.

2.3 HyperNet: Concatenation of multi-scale intermediate outputs

Multi-scale representation and its combination are proven to be effective in many recent deep learning tasks [4, 6, 7]. Combining fine-grained details with highly-abstracted information in feature extraction layer helps the following region proposal network and classification network to detect objects of different scales. However, since the direct concatenation of all abstraction layers may produce redundant information with much higher compute requirement we need to design the number of different abstraction layers and the layer numbers of abstraction carefully. If you choose the layers which are too early for object proposal and classification, it would be little help when we consider additional compute complexity.

Our design choice is not different from the observation from ION [6] and HyperNet [4], which combines 1) the last layer and 2) two intermediate layers whose scales are 2x and 4x of the last layer, respectively. We choose the middle-sized layer as a reference scale (= 2x), and concatenate the 4x-scaled layer and the last layer with down-scaling (pooling) and up-scaling (linear interpolation), respectively.

Name	Type	Stride	Output size	Residual	C.ReLU #1x1-KxK-1x1	Inception #1x1	#3x3	#5x5	#pool	#out	# params	MAC
conv1_1	7x7 C.ReLU	2	528x320x32		X-16-X						2.4K	397M
pool1_1	3x3 max-pool	2	264x160x32									
conv2_1	3x3 C.ReLU		264x160x64	O	24-24-64						11K	468M
conv2_2	3x3 C.ReLU		264x160x64	O	24-24-64						9.8K	414M
conv2_3	3x3 C.ReLU		264x160x64	O	24-24-64						9.8K	414M
conv3_1	3x3 C.ReLU	2	132x80x128	O	48-48-128						44K	468M
conv3_2	3x3 C.ReLU		132x80x128	O	48-48-128						39K	414M
conv3_3	3x3 C.ReLU		132x80x128	O	48-48-128						39K	414M
conv3_4	3x3 C.ReLU		132x80x128	O	48-48-128						39K	414M
conv4_1	Inception	2	66x40x256	O		64	48-128	24-48-48	128	256	247K	653M
conv4_2	Inception		66x40x256	O		64	64-128	24-48-48		256	205K	542M
conv4_3	Inception		66x40x256	O		64	64-128	24-48-48		256	205K	542M
conv4_4	Inception		66x40x256	O		64	64-128	24-48-48		256	205K	542M
conv5_1	Inception	2	33x20x384	O		64	96-192	32-64-64	128	384	573K	378M
conv5_2	Inception		33x20x384	O		64	96-192	32-64-64		384	418K	276M
conv5_3	Inception		33x20x384	O		64	96-192	32-64-64		384	418K	276M
conv5_4	Inception		33x20x384	O		64	96-192	32-64-64		384	418K	276M
downscale	3x3 max-pool	2	66x40x128									
upscale	4x4 deconv	2	66x40x384								6.2K	16M
concat	concat		66x40x768									
convf	1x1 conv		66x40x512								393K	1038M
Total											3282K	7942M

Table 1: The detailed structure of PVANET. All conv layers are combined with batch normalization, channel-wise scaling and shifting, and ReLU activation layers. Theoretical computational cost is given as the number of adds and multiplications (MAC), assuming that the input image size is 1056x640. **KxK C.ReLU** refers to a sequence of "1x1 - KxK - 1x1" conv layers, where KxK is a C.ReLU block as in Figure 1. conv1_1 has no 1x1 conv layer. "C.ReLU" column shows the number of output channels of each conv layer. For **Residual**, 1x1 conv is applied for projecting pool1_1 into conv2_1, conv2_3 into conv3_1, conv3_4 into conv4_1, and conv4_4 into conv5_1. **Inception** consists of four sub-sequences: 1x1 conv (#1x1); "1x1 - 3x3" conv (#3x3); "1x1 - 3x3 - 3x3" conv (#5x5); "3x3 max-pool - 1x1 conv" (#pool, only for stride 2). "#out" refers to 1x1 conv after concatenating those sub-sequences. The number of output channels of each conv layer is shown. **Multi-scale features** are obtained by four steps: conv3_4 is down-scaled into "downscale" by 3x3 max-pool with stride 2; conv5_4 is up-scaled into "upscale" by 4x4 channel-wise deconvolution whose weights are fixed as bilinear interpolation; "downscale", conv4_4 and "upscale" are combined into "concat" by channel-wise concatenation; after 1x1 conv, the final output is obtained (convf).

2.4 Deep network training

It is widely accepted that as network goes deeper and deeper, the training of network becomes more troublesome. We solve this issue by adopting residual structures [1]. Unlike the original residual training idea, we add residual connections onto inception layers as well to stabilize the later part of our deep network architecture.

We also add Batch normalization [5] layers before all ReLU activation layers. Mini-batch sample statistics are used during pre-training, and moving-averaged statistics are used afterwards as fixed scale-and-shift parameters.

Learning rate policy is also important to train network successfully. Our policy is to control the learning rate dynamically, based on plateau detection [1]. We measure the moving average of loss, and decide it to be *on-plateau* if its improvement is below a threshold during a certain period of iterations. Whenever the plateau is detected, the learning rate is decreased by a constant factor. In experiments, our learning rate policy gave a significant gain of accuracy.

3 Faster R-CNN with our feature extraction network

Table 1 shows the whole structure of PVANET. In the early stage (conv1_1, ..., conv3_4), C.ReLU is adapted to convolutional layers to reduce the computational cost of KxK conv by half. 1x1 conv layers are added before and after the KxK conv, in order to reduce the input size and then enlarge the representation capacity, respectively.

Three intermediate outputs from conv3_4 (with down-scaling), conv4_4, and conv5_4 (with up-scaling) are combined into the 512-channel multi-scale output features (convf), which are fed into the Faster R-CNN modules:

- For computational efficiency, only the first 128 channels in convf are fed into the region proposal network (RPN). Our RPN is a sequence of "3x3 conv (384 channels) - 1x1 conv (25x(2+4) = 150 channels[2])" layers to generate regions of interest (RoIs) from
- R-CNN takes all 512 channels in convf. For each RoI, 6x6x512 tensor is generated by RoI pooling, and then passed through a sequence of fully-connected layers of "4096 - 4096 - (21+84)" output nodes.[3]

4 Experimental results

4.1 Training and testing

PVANET was pre-trained with ILSVRC2012 training images for 1000-class image classification.[4] All images were resized into 256x256, and 192x192 patches were randomly cropped and used as the network input. The learning rate was initially set to 0.1, and then decreased by a factor of $1/\sqrt{10} \approx 0.3165$ whenever a plateau is detected. Pre-training terminated if the learning rate drops below $1e-4$, which usually requires about 2M iterations.

Then PVANET was trained with the union set of MS COCO[5] trainval, VOC2007[6] trainval and VOC2012[7] trainval. Fine-tuning with VOC2007 trainval and VOC2012 trainval was also required afterwards, since the class definitions in MS COCO and VOC competitions are slightly different. Training images were resized randomly such that a shorter edge of an image to be between 416 and 864.

For PASCAL VOC evaluations, each input image was resized such that its shorter edge to be 640. All parameters related to Faster R-CNN were set as in the original work [8] except for the number of proposal boxes before non-maximum suppression (NMS) ($= 12000$) and the NMS threshold ($= 0.4$). All evaluations were done on Intel i7-6700K CPU with a single core and NVIDIA Titan X GPU.

4.2 VOC2007

Table 2 shows the accuracy of our models in different configurations. Thanks to Inception (Section 2.2) and multi-scale features (Section 2.3), our RPN generated initial proposals very accurately. Since the results imply that more than 200 proposals does not give notable benefits to detection accuracy, we fixed the number of proposals to 200 in the remaining experiments. We also measured the performance with bounding-box voting [10], while iterative regression was not applied.

Faster R-CNN consists of fully-connected layers, which can be compressed easily without a significant drop of accuracy [11]. We compressed the fully-connected layers of "4096 - 4096" into to "512 - 4096 - 512 - 4096" by the truncated singular value decomposition (SVD), with some fine-tuning after that. The compressed network achieved 81.2% mAP (-0.6%) and ran in 31.3 FPS (+9.6 FPS).

[2] RPN produces 2 predicted scores (foreground and background) and 4 predicted values of the bounding box for each anchor. Our RPN uses 25 anchors of 5 scales (3, 6, 9, 16, 25) and 5 aspect ratios (0.5, 0.667, 1.0, 1.5, 2.0).

[3] For 20-class object detection, R-CNN produces 21 predicted scores (20 classes + 1 background) and 21x4 predicted values of 21 bounding boxes.

[4] http://www.image-net.org/challenges/LSVRC/2012/

[5] http://mscoco.org/dataset/

[6] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

[7] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

Model	Proposals	Recall (%)	mAP (%)	Time (ms)	FPS
PVANET	300	98.9	81.5	48.5	20.6
	200	98.3	81.5	42.2	23.7
	100	97.0	81.3	40.0	25.0
	50	94.7	80.5	26.8	37.3
PVANET+	200	98.3	**81.8**	46.1	21.7
PVANET+ (compressed)	200	98.3	**81.2**	31.9	31.3

Table 2: Performance on VOC2007-test benchmark data. "Recall" refers to a ratio of "true positive (TP)" boxes among the proposals, considering a box as TP if the intersection-over-union (IoU) score with its maximally-overlapped ground-truth box is ≥ 0.5. PVANET+ denotes that bounding-box voting is applied, and PVANET+ (compressed) denotes that fully-connected layers in R-CNN are compressed.

Model	Computation cost (MAC)				Running time		mAP
	Shared CNN	RPN	Classifier	Total	ms	x(PVANET)	(%)
PVANET+	7.9	1.3	27.7	37.0	46	1.0	82.5
Faster R-CNN + ResNet-101	80.5	N/A	219.6	300.1	2240	48.6	83.8
Faster R-CNN + VGG-16	183.2	5.5	27.7	216.4	110	2.4	75.9
R-FCN + ResNet-101	122.9	0	0	122.9	133	2.9	82.0

Table 3: Comparisons between our network and some state-of-the-arts in the PASCAL VOC2012 leaderboard. PVANET+ denotes PVANET with bounding-box voting. We assume that PVANET takes a 1056x640 image and the number of proposals is 200. Competitors' MAC are estimated from their Caffe prototxt files which are publicly available. All testing-time configurations are the same with the original articles [1, 12, 8]. Competitors' runtime performances are also therein, while we projected the original values with assuming that NVIDIA Titan X is 1.5x faster than NVIDIA K40.

4.3 VOC2012

Table 3 summarizes comparisons between PVANET+ and some state-of-the-art networks [1, 8, 12] from the PASCAL VOC2012 leaderboard.[8]

Our PVANET+ achieved 82.5% mAP, the 2nd place on the leaderboard, outperforming all other competitors except for "Faster R-CNN + ResNet-101". However, the top-performer uses ResNet-101 which is much heavier than PVANET, as well as several time-consuming techniques such as global contexts and multi-scale testing, leading to 40x (or more) slower than ours. In Table 3, we also compare mAP with respect to the computational cost. Among the networks performing over 80% mAP, PVANET+ is the only network running ≤ 50ms. Taking its accuracy and computational cost into account, our PVANET+ is the most efficient network in the leaderboard.

5 Conclusions

In this paper, we showed that the current networks are highly redundant and we can design a thin and light network which is capable enough for complex vision tasks. Elaborate adoption and combination of recent technical innovations on deep learning makes us possible to re-design the feature extraction part of the Faster R-CNN framework to maximize the computational efficiency. Even though the proposed network is designed for object detection, we believe our design principle can be widely applicable to other tasks like face recognition and semantic analysis.

Our network design is completely independent of network compression and quantization. All kinds of recent compression and quantization techniques are applicable to our network as well to further increase the actual performance in real applications. As an example, we showed that a simple technique like truncated SVD could achieve a notable improvement in the runtime performance based on our network.

[8] `http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=4`

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun. HyperNet: Towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[6] 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 *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] 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 (NIPS)*, 2015.

[9] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. *arXiv preprint arXiv:1509.09308*, 2015.

[10] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region & semantic segmentation-aware CNN model. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[11] Ross Girshick. Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[12] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn : Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.