An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection

Youngwan Lee* ETRI yw.lee@etri.re.kr
Joong-won Hwang* ETRI jwhwang@etri.re.kr
Sangrok Lee† SK C&C srk@sk.com
Yuseok Bae ETRI ysbae@etri.re.kr
Jongyoul Park ETRI jongyoul@etri.re.kr

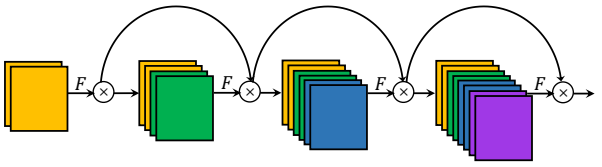
Abstract

As DenseNet conserves intermediate features with diverse receptive fields by aggregating them with dense connection, it shows good performance on the object detection task. Although feature reuse enables DenseNet to produce strong features with a small number of model parameters and FLOPs, the detector with DenseNet backbone shows rather slow speed and low energy efficiency. We find the linearly increasing input channel by dense connection leads to heavy memory access cost, which causes computation overhead and more energy consumption. To solve the inefficiency of DenseNet, we propose an energy and computation efficient architecture called VoVNet comprised of One-Shot Aggregation (OSA). The OSA not only adopts the strength of DenseNet that represents diversified features with multi receptive fields but also overcomes the inefficiency of dense connection by aggregating all features only once in the last feature maps. To validate the effectiveness of VoVNet as a backbone network, we design both lightweight and large-scale VoVNet and apply them to one-stage and two-stage object detectors. Our VoVNet based detectors outperform DenseNet based ones with $2\times$ *faster speed and the energy consumptions are reduced by* $1.6\times$ - $4.1\times$. *In addition to DenseNet, VoVNet also outperforms widely used ResNet backbone with faster speed and better energy efficiency. In particular, the small object detection performance has been significantly improved over DenseNet and ResNet.*

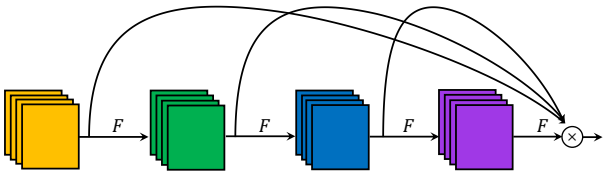
1. Introduction

With the massive progress of convolutional neural networks (CNN) such as VGGNet [23], GoogleNet [25], Inception-V4 [24], ResNet [7], and DenseNet [9], it has become mainstream in object detector to adopt the modern state-of-the-art CNN models as feature extractor. As DenseNet is reported to achieve state-of-the-art performance in the classification task recently, it is natural to attempt to expand its usage to detection tasks. In our experiment (Table 4), we find that the DenseNet based detectors with fewer parameters and FLOPs outperform the detectors with ResNet, which is most widely used for the backbone of object detections.

The main difference between ResNet and DenseNet is the way they aggregate their features; ResNet aggregates the features from shallower by summation while DenseNet does it by concatenation. As mentioned by Zhu et al. [32],



(a) Dense Aggregation (DenseNet)



(b) One-Shot Aggregation (VoVNet)

Figure 1. Aggregation methods. (a) Dense aggregation of DenseNet [9] aggregates all previous features at every subsequent layers, which increases linearly input channel size with only a few new outputs. (b) Our proposed One-Shot Aggregation concatenates all features only once in the last feature map, which makes input size constant and enables enlarging new output channel. F represents convolution layer and $\otimes$ indicates concatenation.

*equal contribution
†This work was done when Sangrok Lee was an intern at ETRI.

information carried by early feature maps would be washed out as it is summed with others. On the other hand, by concatenation, information would last as it preserves original forms. Several works [25, 17, 13] demonstrate that the abstracted feature with multiple receptive fields can capture visual information in various scales. As detection task requires models to recognize an object in more various scale than classification, preserving information from various layers is especially important for detection as each layer has different receptive fields. Therefore, preserving and accumulating feature maps of multiple receptive fields, DenseNet has better and diverse feature representation than ResNet in terms of object detection task.

However, we also find in the experiment that detectors with DenseNet which has fewer FLOPs and model parameters spend more energy and time than those with ResNet. This is because there are other factors than FLOPs and model size that influence on energy and time consumption. First, memory access cost (MAC) required to accessing memory for intermediate feature maps is crucial factor of the consumptions [18, 28]. As illustrated in Figure 1(a), since all previous feature maps in DenseNet are used as input to the subsequent layer by dense connection, it causes the memory access cost to increase quadratically with network depth and in turn leads to computation overhead and more energy consumption.

Second, with respect to GPU parallel computation, DenseNet has the limitation of computation bottleneck. In general, GPU parallel computing utilization is maximized when operand tensor is larger [19, 29, 13]. However, due to linearly increasing input channel, DenseNet is needed to adopt 1×1 convolution bottleneck architecture for reducing input dimension and FLOPs, which results in rather increasing the number of layers with smaller operand tensor. As a result, GPU-computation becomes inefficiency.

The goal of this paper is to improve DenseNet to be more efficient while preserving the benefit from concatenative aggregation for object detection task. We first discuss about MAC and GPU-computation efficiency and how to consider the factors in architecture designing stage. Secondly, we claim that the dense connections in intermediate layers of DenseNet are inducing the inefficiencies and hypothesize that the dense connections are redundant. With these thoughts, we propose a novel One-Shot Aggregation (OSA) that aggregates intermediate features at once as shown in Figure 1(b). This aggregation method brings great benefit to MAC and GPU computation efficiency while it preserves the strength of concatenation. With OSA modules, we build VoVnet[1], energy efficient backbone for real-time detection. To validate the effectiveness of VoVNet as backbone network, we apply VoVNet to various object detectors such as DSOD, RefineDet, and Mask R-CNN. The results show that VoVNet based detectors outperform DenseNet or ResNet based ones with better energy efficiency and speed.

[1]It means Variety of View Network

2. Factors of Efficient Network Design

When designing *efficient* network, many studies such as MobileNet v1 [8], MobileNet v2 [21], ShuffleNet v1 [31], ShuffleNet v2 [18], and Pelee [26] have focused mainly on reducing *FLOPs* and *model sizes* by using depthwise convolution and 1×1 convolution bottleneck architecture. However, reducing FLOPs and model sizes does not always guarantee the reduction of GPU inference time and real energy consumption. Ma *et al*. [18] shows an experiment that ShuffleNet v2 with a similar number of FLOPs runs faster than MobileNet v2 on GPU. Chen *et al*. [2] also shows that while SqueezeNet has 50x fewer weights than AlexNet, it consumes more energy than AlexNet. These phenomena imply that FLOPs and model sizes are indirect metrics to measure practicality and designing the network based on the metrics should be reconsidered. To build *efficient* network architectures that focus on a more practical and valid metrics such as energy per image and frame per second (FPS), besides FLOPs and model parameters, it is important to consider other factors that influence on energy and time consumption.

2.1. Memory Access Cost

The first factor we point out is *memory accesses cost* (MAC). The main source of energy consumption in CNN is memory accesses than computation [28]. Specifically, accessing data from the DRAM (Dynamic Random Access Memory) for an operation consumes orders of magnitude higher energy than the computation itself. Moreover, the time budget on memory access accounts for a large proportion of time consumption and can even be the bottleneck of the GPU process [18]. This implies that even under the same number of computation and parameter if the total number of memory access varies with model structure, the energy consumption will be also different.

One reason that causes the discrepancy between model size and the number of memory access is the *intermediate activation memory footprint*. As stated by Chen *et al*. [1], the memory footprint is attributed to both filter parameter and intermediate feature maps. If the intermediate feature maps are large, the cost for memory access increases even with the same model parameter. Therefore, we consider MAC, which covers the memory footprint for filter parameter and intermediate feature map size both, to an important factor for network design. Specifically, we follow the method of Ma *et al*. [18] to calculate MAC of each convolutional layers as below

$$\text{MAC} = hw(c_i + c_o) + k^2 c_i c_o \quad (1)$$

The notations k, h, w,c_i, c_o denote kernel size, height/width of input and output response, the channel size of input, and that of output response, respectively.

2.2. GPU-Computation Efficiency

The network architectures that reduce their FLOPs for speed is based on the idea that every floating point operation is processed on the same speed in a device. However, this is incorrect when a network is deployed on GPU. This is because of GPU parallel processing mechanism. As GPU is able to process multiple floating processes in time, it is important to utilize its computational ability efficiently. We use the term GPU-computation efficiency for this concept.

GPU parallel computing power is utilized better as the computed data tensor becomes larger [29, 13]. Splitting a large convolution operation into several fragmented smaller operations makes GPU computation inefficient as fewer computations are processed in parallel. In the context of network design, this implies that it is better to compose network with fewer layers if the behavior function is same. Moreover, adopting extra layers causes kernel launching and synchronization which result in additional time overhead [18].

Accordingly, although the technique such as depthwise convolution and 1×1convolution bottleneck can reduce the number of FLOPs, it is harmful to GPU-computation efficiency as it adopts additional 1×1 convolution. More generally, GPU-computation efficiency varies with the model architecture. Therefore, for validating computation efficiency of network architectures, we introduce *FLOPs per Second* (FLOP/s) which is computed by dividing the actual GPU inference time from the total FLOPs. High FLOP/s implies the architecture utilize GPU power efficiently.

3. Proposed Method

3.1. Rethinking Dense Connection

The dense connection that aggregates all intermediate layers induces inevitable inefficiency, which comes from that input channel size of each layer increases linearly as the layer proceed. Because of the intensive aggregation, the dense block can produce only a few features with FLOPs or parameters constraint. In other words, DenseNet trades the quantity of features for the quality of features via the dense connection. Although the performance of DenseNet seems to prove the trade is beneficial, there are some other drawbacks of the trade in perspective of energy and time.

First, dense connections induce high *memory access cost* which is paid by energy and time. As mentioned by Ma *et al*. [18], the lower boundary of MAC, or the number of memory access operation, of a convolutional layer can be represented by $MAC \geq 2\sqrt{\frac{hwB}{k^2}} + \frac{B}{hw}$ when $B = k^2 hwc_i c_o$ is the number of computation. Because the lower

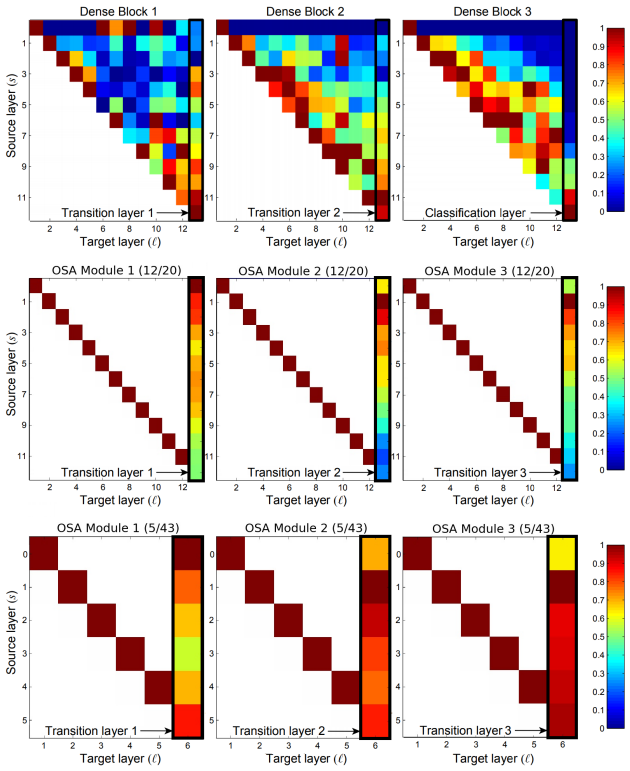


Figure 2. The average absolute filter weights of convolutional layers in trained DenseNet [9] (*top*) and VoVNet (*middle, bottom*). The color of pixel (i, j) encodes the average $L1$ norm of weights connecting layer s to l. OSA module (x/y) indicates that the OSA modules consist of x layers with y channels.

boundary has its ground on mean value inequality, MAC can be minimized when the input and output have the same channel size under fixed number of computation or model parameter. Dense connections increase input channel size while output channel size remains constant, and as a result, each layer has imbalanced input and output channel sizes. Therefore, DenseNet has high MAC among the models with the same number of computations or parameters and consumes more energy and time.

Second, the dense connection imposes the use of bottleneck structure which harms the efficiency of GPU parallel computation. The linearly increasing input size is critically problematic when model size is big because it makes the overall computation grows quadratically with respect to depth. To suppress this growth, DenseNet adopts the bottleneck architecture which adds 1×1 convolutional layers to maintain the input size of 3×3 convolutional layer constant. Although this solution can reduce FLOPs and parameters, it harms the GPU parallel computation efficiency as discussed. Bottleneck architecture divides one 3×3 convolutional layer into two smaller layers and causes more sequential computations, which lowers the inference speed.

Because of these drawbacks, DenseNet becomes ineffi-

cient in terms of energy and time. To improve efficiency, we first investigate how dense connections actually aggregate the features once the network is trained. Hu *et al*. [9] illustrate the connectivity of the dense connection by evaluating normalized L1 norm of input weights to each layer. These values show the normalized influences of each preceding layer to corresponding layers. The figures are represented in Figure 2 (*top*).

In Dense Block3, the red boxes near the diagonal show that aggregations on intermediate layers are active. However, in the classification layer, only a small proportion of intermediate features is used. In contrast, in Dense Block1 transition layer aggregates the most of its input feature well while intermediate layers do not.

With the observations, we hypothesize that there is a negative relation between the strength of aggregation on intermediate layers and that of final layers. This can be true if the dense connection between intermediate layers induces correlation between features from each layer. This means that dense connection makes later intermediate layer produce the features that are better but also similar to the features from former layers. In this case, the final layer is not required to learn to aggregate both features because they are representing *redundant information*. As a result, the influence of the former intermediate layer to the final layer becomes small.

As all intermediate features are aggregated to produce final feature in the final layer, it is better to produce intermediate features that can complement each other, or less correlated. Therefore, we can extend our hypothesis to that the effect of dense connections in intermediate feature is relatively *little* with respect to the cost. To verify the hypotheses, we redesign a novel module that aggregates its intermediate features only on the final layer of each block.

3.2. One-Shot Aggregation

We integrate previously discussed thoughts into efficient architecture, one-shot aggregation (OSA) module which aggregates its feature in the last layer at once. Figure 1(b) illustrates the proposed OSA module. Each convolution layer is connected by two-way connection. One way is connected to the subsequent layer to produce the feature with a larger receptive field while the other way is aggregated only once into the final output feature map. The difference with DenseNet is that the output of each layer is not routed to all subsequent intermediate layers which makes the input size of intermediate layers constant.

To verify our hypotheses that there is a negative relation between the strength of aggregation on intermediate layers and that on final layer, and that the dense connections are redundant, we conduct the same experiment with Hu *et al*. [9] on OSA module. We designed OSA modules to have the similar number of parameter and computation with dense block which is used in DenseNet-40. First, we investigate the result on the OSA module with the same number of layers with the dense block, which is 12 (Figure 2 (*middle*)). The output is bigger than that of dense block as the input size of each convolution layers is reduced. The network with OSA modules shows 93.6% accuracy on CIFAR-10 classification which is slightly dropped by 1.2% but still higher than ResNet with similar model size. It can be observed that the aggregations in final layers become more intense as the dense connections on intermediate layers are pruned.

Moreover, the weights of transition layer of OSA module show the different pattern with that of DenseNet: features from shallow depth are more aggregated on the transition layer. Since the features from deep layer are not influencing strongly on transition layers, we can reduce the layer without significant effect. Therefore, we reconfigure OSA module to have 5 layers with 43 channels each (Figure 2 (*bottom*)). Surprisingly, with this module, we achieve error rate 5.44% which is similar to that of DenseNet-40 (5.24%). This implies that building deep intermediate feature via dense connection is less effective than expected.

Although the network with OSA module has slightly decreased performance on CIFAR-10, which does not necessarily imply it will underperform on detection task, it has much less MAC than that with dense block. By following Eq. (1), it is estimated that substituting dense block of DenseNet-40 to OSA module with 5 layers with 43 channels reduces MAC from 3.7M to 2.5M. This is because the intermediate layers in OSA have the same size of input and output which leads MAC to the lower boundary. This means that one can build faster and more energy efficient network if the MAC is the dominant factor of energy and time consumption. Specifically, as detection is performed on a higher resolution than classification, the intermediate memory footprint will become larger and MAC will reflect the energy and time consumption more appropriately.

Also, OSA improves GPU computation efficiency. The input sizes of intermediate layers of OSA module are constant. Hence, it is unnecessary to adopt additional 1×1 conv bottleneck to reduce dimension. Moreover, as the OSA module aggregates the shallow features, it consists of fewer layers. As a result, the OSA module is designed to have only a few layers that can be efficiently computed in GPU.

3.3. Configuration of VoVNet

Due to the diversified feature representation and efficiency of the OSA modules, our VoVNet can be constructed by stacking only a few modules with high accuracy and fast speed. Based on the confirmation that the shallow depth is more aggregated in Figure 2, we can configure the OSA module with a smaller number of convolutions with larger channel than DenseNet. There are two types of VoVNet:

Type	Output Stride	VoVNet-27-slim	VoVNet-39	VoVNet-57
Stem	2	3×3 conv, 64, stride=2	3×3 conv, 64, stride=2	3×3 conv, 64, stride=2
Stage 1	2	3×3 conv, 64, stride=1	3×3 conv, 64, stride=1	3×3 conv, 64, stride=1
	2	3×3 conv, 128, stride=1	3×3 conv, 128, stride=1	3×3 conv, 128, stride=1
OSA module Stage 2	4	$\begin{bmatrix} 3 \times 3 \text{ conv, } 64, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 128 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3 \text{ conv, } 128, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 256 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3 \text{ conv, } 128, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 256 \end{bmatrix} \times 1$
OSA module Stage 3	8	$\begin{bmatrix} 3 \times 3 \text{ conv, } 80, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 256 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3 \text{ conv, } 160, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 512 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3 \text{ conv, } 160, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 512 \end{bmatrix} \times 1$
OSA module Stage 4	16	$\begin{bmatrix} 3 \times 3 \text{ conv, } 96, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 384 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3 \text{ conv, } 192, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 768 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \text{ conv, } 192, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 768 \end{bmatrix} \times 4$
OSA module Stage 5	32	$\begin{bmatrix} 3 \times 3 \text{ conv, } 112, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 512 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3 \text{ conv, } 224, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 1024 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \text{ conv, } 224, \times 5 \\ \text{concat \& } 1\times1 \text{ conv, } 1024 \end{bmatrix} \times 3$

Table 1. Overall architecture of VoVNet. Downsampling is done by 3×3 max pooling with a stride of 2 at the end of each stage. Note that each *conv* layer has the sequence Conv-BN-ReLU.

lightweight network, *e.g.*, VoVNet-27-slim, and large-scale network, *e.g.*, VoVNet-39/57. VoVNet consists of a stem block including 3 convolution layers and 4 stages of OSA modules with output stride 32. An OSA module is comprised of 5 convolution layers with the same input/output channel for minimizing MAC as discussed in Section 3.1. Whenever the stage goes up, the feature map is downsampled by 3×3 max pooling with stride 2. VoVNet-39/57 have more OSA modules at the 4th and 5th stage where downsampling is done in the last module.

Since the semantic information in high-level is more important for object detection task, we increase the proportion of high-level features relative to low-level ones by growing the output channels at different stages. Contrary to the limitation of only a few new outputs in DenseNet, our strategy allows VoVNet to express better feature representation with fewer total layers (*e.g.*, VoVNet-57 vs. DenseNet-161). The details of VoVNet architecture are shown in Table 1.

4. Experiments

In this section, we validate the effectiveness of the proposed VoVNet as backbone for object detection in terms of GPU-computation and energy efficiency. At first, for comparison with lightweight DenseNet, we apply our lightweight VoVNet-27-slim to DSOD [22] that is the first detector using DenseNet. Then, we compare with state-of-the-art *lightweight* object detectors such as Pelee [26] that also uses a DenseNet-variant backbone and SSD-MobileNet [8].

Furthermore, to validate the possibility of generalization to large-scale models, we extend the VoVNet to state-of-the-art one-stage detector, *e.g.*, RefineDet [30], and two-stage detector, *e.g.*, Mask R-CNN [6], on more challenging COCO [16] dataset. Since ResNet is the most widely used backbone for object detection and segmentation task, we compare VoVNet with ResNet as well as DenseNet. In particular, we compare the speed and accuracy of VoVNet-39/57 with DenseNet-201/161 and ResNet-50/101 as they have similar model sizes.

4.1. Experimental setup

Speed Measurement. For fair speed comparison, we measure the inference time of all models in Table 2, 4 on the same GPU workstation with TITAN X GPU (Pascal architecture), CUDA v9.2, and cuDNN v7.3. It is noted that Pelee [26] merges batch normalization layer into convolution for accelerating the inference time. As the other models also have batch normalization layers, we compare Pelee without merge-bn trick for fair comparison.

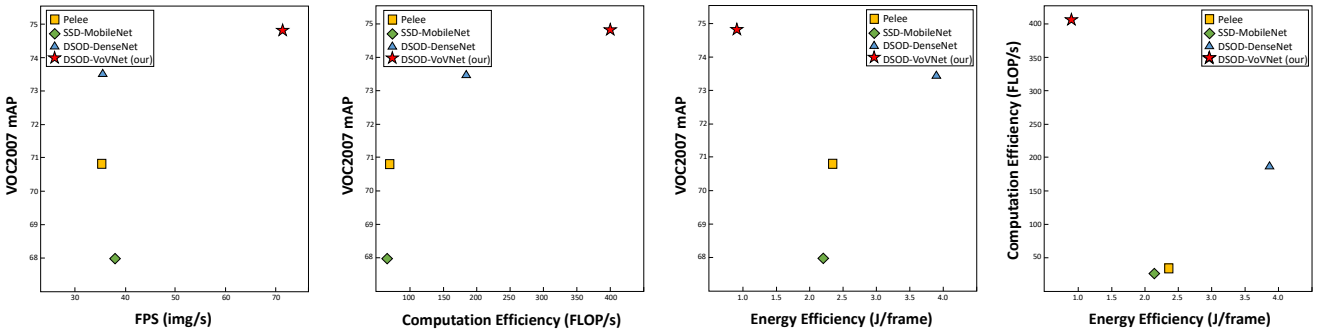
Energy Consumption Measurement. We measure the energy consumption of both lightweight and large-scale models during object detection evaluation of VOC2007 `test` images (*e.g.*, 4952 images) and COCO `minival` images (*e.g.*, 5000 images), respectively. GPU power usage is measured with Nvidia's system monitor interface (`nvidia-smi`). We sample the power value with an interval of 100 millisecond and compute average of the measured power. The energy consumption per image can be calculated as below

$$\frac{\text{Average Power [Joule/Second]}}{\text{Inference speed [Frame/Second]}} \tag{2}$$

We also measure total memory usage that includes not only model parameters but also intermediate activation maps. The measured energy and memory footprint in Table 2.

4.2. DSOD

To validate the effectiveness of backbone part, except for replacing DenseNet-67 (referred to DSOD [22] as DS-64-64-16-1) with our VoVNet-27-slim, we follow the same hyper-parameters such as default box scale, aspect ratio, and dense prediction and the training protocol such as 128 total batch size, 100k max iterations, initial learning rate, and learning rate schedule. DSOD with VoVNet is trained on the union of VOC2007 `trainval` and VOC2012 `trainval`("07+12") following [22]. As the original DSOD with DenseNet-67 is trained from scratch, we also train our model without ImageNet pretrained



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

Figure 3. Comparisons of lightweight models in terms of the computation and energy efficiency. (a) shows speed vs. accuracy. (b), (c), and (d) illustrate comparison of GPU-computation-efficiency, energy-efficiency and GPU-computation vs. energy efficiency, respectively.

Detector	Backbone	FLOPs (G)	FPS (img/s)	#Param (M)	Memory footprint (MB)	Energy Efficiency (J/img)	Computation Efficiency (GFLOP/s)	mAP
SSD300	MobileNet [8]	1.1	37	5.7	766	2.3	42	68.0
Pelee304	PeleeNet [26]	1.2	35	5.4	1104	2.4	43	70.9
DSOD300	DenseNet-67 [22]	5.3	35	5.9	1294	3.7	189	73.6
DSOD300	**VoVNet-27-slim**	5.6	71	5.9	825	0.9	400	74.8

Table 2. Comparison with lightweight object detectors. All models are trained on VOC 2007 and VOC 2012 *trainval* set and tested on VOC 2007 *test* set.

Backbone	FLOPs (G)	GPU time (ms)	#Param (M)	Memory footprint (MB)	mAP
VoVNet-27-slim	5.6	14	5.9	825	74.8
+ w/ bottleneck	4.6	18	4.8	895	71.1

Table 3. Ablation study on 1×1 convolution bottleneck.

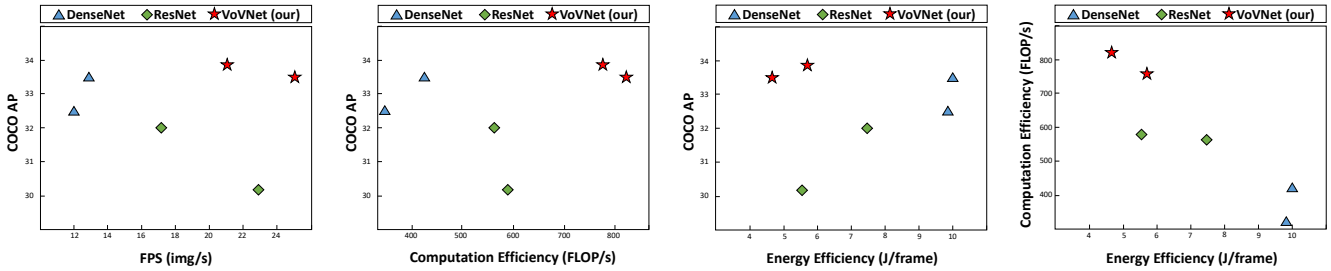
model. We implement DSOD with VoVNet-27-slim based on DSOD original Caffe code[2].

VoVNet vs. DenseNet. As shown in Table 2, the proposed VoVNet-27-slim based DSOD300 achieves 74.87%, which is better than DenseNet-67 based one even with comparable parameters. In addition to accuracy, the inference speed of VoVNet-27-slim is also two times faster than that of the counterpart with comparable FLOPs. The Pelee [26], DenseNet-variant backbone, is designed to decompose a dense block into a smaller two-way dense block, which reduces FLOPs to about $\times 5$ less than DenseNet-67. However, despite the fewer FLOPs, Pelee has similar inference speed with DSOD with DenseNet-67. We conjecture that decomposing a dense block into smaller fragmented layers deteriorates GPU computing parallelism. The VoVNet-27-slim based DSOD also outperforms Pelee by a large margin of 3.97% at much faster speed.

Ablation study on 1×1 conv bottleneck. To check the influence of 1×1 convolution bottleneck on model-efficiency, we conduct an ablation study where we add a 1×1 convolution in front of every 3×3 convolution operation in OSA module with half channel of the input. Table 3 shows comparison results. VoVNet with 1×1 bottleneck reduces FLOPs and the number of model parameters, but conversely increases GPU inference time and memory footprint compared to without one. The accuracy also drops by 3.69% mAP. This is the problem in the same context as why Pelee is slower than DenseNet-67 despite the fewer FLOPs. As the 1×1 bottleneck decomposes a large 3×3 convolution tensor into several smaller tensors, it rather hampers GPU parallel computations. Although the 1×1 bottleneck decreases the number of parameters, it increases the total number of layers in the network which requires more intermediate activation maps and in turn increases overall memory footprint.

GPU-Computation Efficiency. Although SSD-MobileNet and Pelee have much fewer FLOPs compared to DSOD-DenseNet-67, DenseNet-67 shows comparable inference speed on GPU. In addition, even with similar FLOPs, VoVNet-27-slim runs twice as fast as DenseNet-67. These results suggest that FLOPs can not sufficiently reflect the inference time as GPU-computation efficiencies of models differ significantly. Thus, we set FLOP/s, which means how well the network utilizes GPU computing resources, as GPU-computation efficiency. From this valid metric, VoVNet-27-slim achieves the highest 400 GFLOP/s among other methods as described in Figure 3(b). The computation efficiency of VoVNet-27-slim is about $10\times$ higher than those of MobileNet and Pelee, which demonstrates that

[2]https://github.com/szq0214/DSOD



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

Figure 4. Comparisons of large-scale models on RefineDet320 [30] in terms of the computation and energy efficiency. (a) shows speed vs. accuracy. (b), (c), and (d) illustrate comparison of GPU-computation-efficiency and energy-efficiency, respectively.

Backbone	FLOPs (G)	FPS (img/s)	#param (M)	Memory footprint (MB)	Energy Efficiency (J/img)	Computation Efficiency (GFLOP/s)	COCO AP AP/AP_S/AP_M/AP_L
ResNet-50 [7]	25.43	23.2	63.46	2229	5.3	591.3	30.3/10.2/32.8/46.9
DenseNet-201 (k=32) [9]	**24.65**	12.0	**56.13**	3498	9.9	296.9	32.5/11.3/35.4/**50.1**
VoVNet-39	32.6	**25.0**	56.28	**2199**	**4.8**	**815.0**	**33.5/12.8/36.8**/49.2
ResNet-101 [7]	33.02	17.5	82.45	3013	7.5	579.2	32.0/10.5/34.7/50.4
DenseNet-161 (k=48) [9]	**32.74**	12.8	**66.76**	3628	10.0	419.7	33.5/11.6/36.6/**51.4**
VoVNet-57	36.45	**21.2**	70.32	**2511**	**5.9**	**775.5**	**33.9/12.8/37.1**/50.3

Table 4. Comparison backbone networks on RefineDet320 [30] on COCO *test-dev* set.

the depthwise convolution and decomposing a convolution into the smaller fragmented operations are not an efficient way in terms of GPU computation-efficiency. Given these results, it is worth noting that VoVNet makes full use of GPU computation resource most efficiently. As a result, VoVNet achieves a significantly better speed-accuracy tradeoff as shown in Figure 3(a).

Energy Efficiency. When validating the efficiency of network, another important thing to consider is energy efficiency (Joule/frame). The metric is the amount of energy consumed to process an image; the lower value means better energy efficiency. We measure energy consumption and obtain the energy efficiencies of VoVNet and other models based detectors. Table 2 shows a tendency between energy efficiency and memory footprint. VoVNet based DSOD consumes only 0.9J per image, which is 4.1× less than DenseNet based one. We can note that the excessive intermediate activation maps of DenseNet increase the memory footprint, which results in more energy consumption. It is also notable that MobileNet shows worse energy efficiency than VoVNet although its memory footprint is lower. This is because depthwise convolution requires fragmented memory access and in turn increases memory access costs [11].

Figure 3(c) describes *accuracy vs. energy efficiency* where with two times better energy efficiency than MobileNet and Pelee, VoVNet outperforms the counterparts by a large margin of 6.87% and 3.97%, respectively. In addition, Figure 3(d) shows a tendency of efficiency with respect to computation and energy consumption both. VoVNet is located in the left-upper direction, which means it is the most efficient model in terms of both GPU-computation and energy efficiency.

4.3. RefineDet

From this section, we validate the generalization to large-scale VoVNet, *e.g.*,VoVNet-39/57, in RefineDet [30] which is the state-of-the-art one-stage object detector. Without any bells-and-whistles, we simply plug VoVNet-39/57 into RefineDet, following same hyper-parameters and training protocols for fair comparison. We train RefineDet320 for 400k iterations with a batch size of 32 and an initial learning rate of 0.001 which is decreased by 0.1 at 280k and 360k iterations. All models are implemented by RefineDet original Caffe code[3] base. The results are summarized in Table 4.

Accuracy vs. Speed. Figure 4(a) illustrates *speed vs. accuracy*. VoVNet-39/57 outperform DenseNet-201/161 and ResNet50/101 both with faster speed. While VoVNet-39 achieves similar accuracy of 33.5 AP with DenseNet-161, it runs about two times faster than the counterpart with much fewer parameters and less memory footprint. VoV-39 also outperforms ResNet-50 by a large margin of 3.3% absolute AP at comparable speed. These results demonstrate with fewer parameters and memory footprint, the proposed VoVNet is the most efficient backbone network in terms of both accuracy and speed.

[3]https://github.com/sfzhang15/RefineDet

GPU-Computation Efficiency. Figure 4(b) shows that VoVNet-39/57 outperform DenseNet and ResNet backbones with higher computation efficiency. In particular, since VoVNet-39 runs faster than DenseNet-201 having fewer FLOPs, VoVNet-39 achieves about three times higher computation efficiency than DenseNet-201 with better accuracy. One can note that although DenseNet-201 (k=32) has fewer FLOPs, it runs slower than DenseNet-161 (k=48), which means lower computation efficiency. We speculate that deeper and thinner network architecture is computationally in-efficient in terms of GPU parallelism.

Energy Effficiency. As illustrated in Figure 4(c), with higher or comparable accuracy, VoV-39/57 consume only 4.8J and 5.9J per image, which are less than DenseNet-201/161 and ResNet-50/101, respectively. Compared to DenseNet161, the energy consumption of VoVNet-39 is two times less with comparable accuracy. Table 4 shows that the positive relation between memory footprint and energy consumption. From this observation, it can be seen that VoVNet with relatively fewer memory footprint is the most energy efficient. In addition, Figure 4(d) shows that our VoVNet-39/57 are located in the most efficient position in terms of energy and computation.

Small Object Detection. In Table 4, we find that VoVNet and DenseNet obtain higher AP than ResNet on small and medium objects. This supports that conserving the diverse feature representations with multi-receptive fields by concatenative aggregation has the advantage of small object detection. Furthermore, VoVNet improves 1.9%/1.2% small object AP gain from DenseNet121/161, which suggests that generating more features by OSA is better than generating deep features by dense connection on small object detection.

4.4. Mask R-CNN from scratch

In this section, we also validate the efficiency of VoVNet as a backbone for a two-stage object detector, Mask R-CNN. Recent works [22, 5] are studied on training without ImageNet pretraining. DSOD is the first one-stage object detector trained from scratch and achieves significant performance due to the deep supervision trait of DenseNet. He *et al*. [5] also prove that when trained from scratch for longer training iterations, Mask R-CNN with Group normalization (GN) [27] achieves comparable or higher accuracy than that with ImageNet pretraining. We also already confirmed our VoVNet with DSOD achieves good performance when training from scratch in Section 4.2.

Thus we also apply VoVNet backbone to Mask R-CNN with GN, the state-of-the-art two-stage object detection and simultaneously instance segmentation. For fair comparison, without any bells-and-whistles, we only exchange ResNet with GN backbone for VoVNet with GN in Mask R-CNN, following same hyperparameters and training protocols [4]. We train VoVNet with GN based Mask R-CNN from scratch with batch size 16 for 3× schedule in an end-to-end manner as like [27]. Meanwhile, due to extreme memory footprint of DenseNet and larger input size of Mask R-CNN, we cannot train DenseNet based Mask R-CNN even on the 32GB V100 GPUs. The results are listed in Table 5.

Backbone	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{70}	AP^{seg}	AP^{seg}_{50}	AP^{seg}_{75}	GPU time
ResNet-50-GN	39.5	59.8	43.6	35.2	56.9	37.6	157 ms
ResNet-101-GN	41.0	61.1	44.9	36.4	58.2	38.7	185 ms
VoVNet-39-GN	41.7	62.2	45.8	36.8	59.0	39.5	152 ms
VoVNet-57-GN	41.9	62.1	46.0	37.0	59.3	39.7	159 ms

Table 5. Detection and segementation results using Mask R-CNN with **Group Normalization** [27] trained ***from scratch*** for 3× schedule and evaluted on COCO *val* set.

Accuracy vs. Speed. For object detection task, with faster speed, VoVNet-39 obtains 2.2%/0.9% absolute AP gains compared to ResNet-50/101, respectively. The extended version of VoVNet, VoVNet-57 also achieves state-of-the-art performance compared to ResNet-101 at faster inference speed. For instance segmentation task, VoVNet-39 also improves 1.6%/0.4% AP from ResNet-50/101. These results support the fact that VoVNet can also provide better diverse feature representation for object detection and simultaneously instance segmentation *efficiently*.

5. Conclusion

For real-time object detection, in this paper, we propose an efficient backbone network called VoVNet that makes good use of the diversified feature representation with multi receptive fields and improves the inefficiency of DenseNet. The proposed One-Shot Aggregation (OSA) addresses the problem of linearly increasing the input channel of the dense connection by aggregating all features in the final feature map only at once. This results in constant input size which reduces memory access cost and makes GPU-computation more efficient. Extensive experimental results demonstrate that not only lightweight but also large-scale VoVNet based detectors outperform DenseNet based ones at much faster speed. For future works, we have plans to apply VoVNet to other detection meta-architectures or semantic segmentation, keypoints detection, etc.

6. Acknowledgement

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (B0101-15-0266, Development of High Performance Visual BigData Discovery platform)

References

[1] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 367–379. IEEE Press, 2016. 2

[2] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Understanding the limitations of existing energy-efficient design approaches for deep neural networks. *Energy*, 2(L1):L3, 2018. 2

[3] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017. 10

[4] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018. 8

[5] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. *arXiv preprint arXiv:1811.08883*, 2018. 8

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 5, 10

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 7, 10

[8] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 5, 6

[9] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1, 3, 4, 7

[10] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 10

[11] Yunho Jeon and Junmo Kim. Constructing fast network through deconstruction of convolution. In *NIPS*, pages 5955–5965, 2018. 7

[12] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, pages 734–750, 2018. 10

[13] Youngwan Lee, Hueun Kim, Eunsoo Park, Xuenan Cui, and Hakil Kim. Wide-residual-inception networks for real-time object detection. In *IV*, pages 758–764. IEEE, 2017. 2, 3

[14] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 10

[15] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *ICCV*, 2017. 10

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5

[17] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *ECCV*, 2018. 2

[18] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 2, 3

[19] Nvidia. Gpu-based deep learning inference: A performance and power analysis. *Nvidia Whitepaper*, 2015. 2

[20] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 10

[21] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv preprint arXiv:1801.04381*, 2018. 2

[22] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *ICCV*, 2017. 5, 6, 8, 10

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[24] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 1

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 2

[26] Robert J Wang, Xiang Li, and Charles X Ling. Pelee: A real-time object detection system on mobile devices. In *NIPS*, pages 1963–1972, 2018. 2, 5, 6

[27] Yuxin Wu and Kaiming He. Group normalization. *arXiv preprint arXiv:1803.08494*, 2018. 8

[28] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *CVPR*, pages 5687–5695, 2017. 2

[29] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 2, 3

[30] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018. 5, 7, 10

[31] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 2

[32] Ligeng Zhu, Ruizhi Deng, Michael Maire, Zhiwei Deng, Greg Mori, and Ping Tan. Sparsely aggregated convolutional networks. In *ECCV*, 2018. 1

Method	Backbone	Input size	Multi Scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
two-stage detectors:										
Faster R-CNN by G-RMI [10]	Inception-ResNet-v2	∼1000×600	False	34.7	55.5	36.7	13.5	38.1	50.8	-
Faster R-CNN+++ [7]	ResNet-101-C4	∼1000×600	False	34.9	55.7	37.4	15.6	38.7	50.9	0.3
Faster R-CNN w FPN [14]	ResNet-101-FPN	∼1000×600	False	36.2	59.1	39	18.2	39	48.2	-
Faster R-CNN, RoIAlign [6]	ResNet-101-FPN	∼1000×600	False	37.3	59.6	40.3	19.8	40.2	48.8	9.2
Mask R-CNN [6]	ResNeXt-101-FPN	∼1280×800	False	39.8	62.3	43.4	22.1	43.2	51.2	5.3
one-stage detectors:										
DSOD300 [22]	DS/64-192-48-1	300×300	False	29.3	47.3	30.6	9.4	31.5	47	28.6
SSD320	ResNet-50	320×320	False	24.9	42.6	25.8	6.9	26.7	41.3	29.4
SSD321 [3]	ResNet-101	321×321	False	28	46.1	29.2	6.2	28.3	49.3	22.7
RefineDet320 [30]	VGG-16	320×320	False	29.4	49.2	31.3	10.0	32.0	44.4	38.7
RefineDet320 [30]	ResNet-50	320×320	False	30.3	49.8	32.3	10.2	32.8	46.9	23.2
RefineDet320 [30]	ResNet-101	320×320	False	32	51.4	34.2	10.5	34.7	50.4	17.5
RefineDet320 [30]	DenseNet-201	320×320	False	32.5	52.2	34.7	11.3	35.4	50.1	12.0
RefineDet320 [30]	DenseNet-161	320×320	False	33.5	53.5	36.0	11.6	36.6	51.4	12.8
RefineDet320 [30]	**VoVNet-39 (ours)**	320×320	False	33.5	53.8	35.8	12.8	36.8	49.2	25.0
RefineDet320 [30]	**VoVNet-57 (ours)**	320×320	False	33.9	54.1	36.3	12.8	37.1	50.3	21.2
YOLOv3-608 [20]	DarkNet-53	608×608	False	33	57.9	34.4	18.3	35.4	41.9	19.6
SSD513 [3]	ResNet-101	513×513	False	31.2	50.4	33.3	10.2	34.5	49.8	13.9
DSSD513 [3]	ResNet-101	513×513	False	33.2	53.3	35.2	12	35.4	51.1	-
RetinaNet500 [15]	Res-101-FPN	500×500	False	34.4	53.1	36.8	14.7	38.5	49.1	11.1
RetinaNet800 [15]	Res-101-FPN	800×800	False	37.8	57.5	40.8	20.2	41.1	49.2	5.0
RefineDet512 [30]	ResNet-101	512×512	False	36.4	57.5	39.5	16.6	39.9	51.4	12.7
RefineDet512+ [30]	ResNet-101	512×512	True	41.8	62.9	45.7	25.6	45.1	54.1	-
CornerNet [12]	Hourglass	512×512	False	40.6	56.4	43.2	19.1	42.8	54.3	4.4
CornerNet [12]	Hourglass	512×512	True	42.2	57.8	45.2	20.7	44.8	**56.6**	-
RefineDet512 [30]	**VoVNet-39 (ours)**	512×512	False	38.5	60.4	42.0	20.0	41.4	51.7	16.6
RefineDet512 [30]	**VoVNet-57 (ours)**	512×512	False	39.2	60.7	42.6	20.2	42.4	52.8	14.9
RefineDet512 [30]	**VoVNet-39 (ours)**	512×512	True	43.0	64.5	46.9	26.8	46.0	54.8	-
RefineDet512 [30]	**VoVNet-57 (ours)**	512×512	True	**43.6**	**64.9**	**47.7**	**27.2**	**46.9**	55.6	-

Table 6. Bechmark results on COCO *test-dev* set.

7. Appendix A: Experiments on RefineDet512

To benchmark VoVNet in RefineDet with larger input size of 512×512, following the same hyper parameters and training protocol [30] as RefineDet512 with ResNet101, we train VoVNet-39/57 based RefineDet512 with a batch size of 20 and an intial learning rate of 10^{-3} for the first 400k iterations, then 10^{-4} and 10^{-5} for another 80k and 60k iterations on COCO dataset. It is noted that DenseNet-201/161 based RefineDet512 cannot be trained due to their heavy memory access cost on 4 NVIDIA V100 GPUs with 32GB.

Table 7 demonstrates RefineDet-VoV39/57 outperform ResNet-50/101 counterparts by margins of 2.3% and 1.7% with better speed, respectively. Furthermore, due to memory-efficiency of VoVNet, We can enlarge batch size from 20 to 32 and train models 400k iterations with initial learning rate of 10^{-3} which decayed by 0.1 at 280k and 360kk iterations. we note that RefineDet512 with ResNet-101 cannot be trained with batch 32 due to its exhausted memory access cost. As described in Table 7, larger batch size leads to absolute 1.0%/1.1% AP gain of VoVNet-39/57.

Backbone	$AP_{20\ batch}$	$AP_{32\ batch}$	FPS
ResNet-50	35.2	-	15.6
ResNet-101	36.4	-	12.3
VoVNet-39	37.5	38.5	16.6
VoVNet-57	38.1	39.2	14.9

Table 7. Comparisons of RefineDet512 on COCO `test-dev`. $AP_{20\ batch}$ and $AP_{32\ batch}$ denote Avearage Precision w.r.t. batch size of 20 and 32, respectively.

Table 6 shows state-of-the-art methods including one-stage and two-stage detectors both. Although RefineDet512 with VoVNet-57 obtains slightly lower accuracy than CornerNet, it runs 3× faster than the counterpart. With multi-scale testing, our VoVNet-57 based RefineDet achieves state-of-the art accuracy over all one-stage and two-stage object detectors.

8. Appendix B: Qualitative comparisons

We display qualitative results on COCO `minival` dataset. In the Figure 5, the detection results of Re-

Figure 5. Comparison of Qualitative detection results. We compare VoVNet-57 with DenseNet-161 and ResNet-101 by combining RefineDet320. The images are from COCO `minival` dataset. Compared to its counterparts, VoVNet-57 can detect small objects better.

fineDet320 based on DenseNet-161, ResNet-101, and VoVNet-57 are compared. The boxes in the figure is bounding boxes that have confidence scores over 0.6. It can be found that the detector with VoVNet outperforms its counterparts. We note that VoVNet is especially strong when objects are small.