Rethinking Convolutional Feature Extraction for Small Object Detection

Burhan A. Mudassar
burhan.mudassar@gatech.edu
Saibal Mukhopadhyay
saibal@ece.gatech.edu

Georgia Institute of Technology
Atlanta, GA 30332, USA

Abstract

Deep learning based object detection architectures have significantly advanced the state of the art. However, a study of recent detection methods shows a wide gap between small object performance and performance on medium and large objects. This gap is prevalent across architectures and across backbones. We show that this gap is primarily due to reduction in the feature map size as we traverse the backbone. Through simple modifications to the backbone structure, we show a marked improvement in performance for small objects. In addition, we propose a dual-path configuration with weight sharing for recovering large object performance. Compared to state of the art methods that rely on multi-scale training and network partitioning we show competitive performance without any bells and whistles on the MS COCO dataset. We show state of the art small object performance with a mobile object detector SSD Mobilenet v1.

1 Introduction

Object detection has made tremendous strides through convolutional neural network based architectures in recent years [2, 3, 4, 8, 10, 11, 12, 13, 15] . Despite these advances, a major hurdle to robust object detection is performance on small objects [6, 15]. An examination of performance metrics for many detectors yields one conclusion; that there is a major difference between detection performance of small and large objects. This gap is present irrespective of the architecture or the backbone. For mobile-friendly detectors the performance gap is even larger [5, 11, 16]. This poses a problem for critical latency sensitive applications that rely on edge-based detection e.g. autonomous driving. As autonomous systems find widespread adoption, it is critical to address this challenge especially in light of recent fatal crashes involving autonomous cars [1].

What makes small object detection difficult is the lack of distinctive features in the object itself. As we go deeper into the convolutional pipeline, the background features mix with and dominate the salient features of small objects. Predicting at an earlier layer does not increase the small object performance as the semantic features are not strong enough for an effective prediction. The current accepted practice is to use multiple scales of an image and do multiple passes over an image possibly with different parameters for detecting small and large objects [6, 15]. This becomes a computationally expensive and impractical solution for detection at the edge.

In this paper, we present a detailed analysis of recent mobile detection methods on small, medium and large objects. While previous studies [6, 15] have shown that pre-training bias affects the performance at difference scales, we argue that it as also affected by the design principles employed for object detection. Instead of designing a custom backbone for object detection [7, 17], we show that we can make changes to existing backbones without resorting to training from scratch (despite pretraining). In addition, we show that these changes impact accuracy for large objects. Contrary to previous work that adds additional layers for large object detection [7, 14, 17], we propose a dual-path weight sharing scheme with separate paths for small and large objects. We implement our approach on a simple mobile object detector SSD mobilenet v1 and show increased performance on small objects. Compared to the base configuration, we improve Average Recall (AR) by 14.8% and Average Precision (AP) by 6.4% for small objects on the MS COCO 2017 validation dataset . In addition, we show similar improvements with more complex backbones such as VGG-16. Our proposed approach is parameter efficient using the same amount of parameters for VGG-16 as the base configuration. Compared to RFB300 [11], our VGG 16 configuration shows higher small AP/AR (13.7% vs 11.9% / 30.4% vs. 19.1%) with a smaller memory footprint (35.1 MParams vs. 45.1 MParams)

2 Related Work

Currently, there are two popular convnet-based approaches to object detection. Two-stage detection methods localize first followed by classification and regression of bounding boxes. A region proposal network (RPN) generates proposals for likely locations of objects using the convolutional feature maps. ROI pooling then uses these proposals to max pool object-specific features for classification. Faster R-CNN [13] and RFCN [7] are two popular variants of this approach. Single shot detectors classify and localize in the same step. Single-Shot MultiBox Detector (SSD) is the most popular variant [12]. It uses several intermediate feature maps and fully convolution layers to perform prediction. Multiple variants of SSD have been proposed. RetinaNet [10] adds weighted cross entropy loss for suppressing easily classified negative examples. RFB Net [11] replaces the top convolution layers in SSD with Inception-like units with multi-scale filters. FSSD [8] concatenates the intermediate feature maps and uses a separate deep neural net for prediction.

Small Object Detection: Multiple recent works have concentrated on dealing with scale invariant object detection and particularly object detection for small objects. In summary, these approaches include training and testing at multiple scales [6, 15], creating feature pyramids [6, 9] and modifying training strategies [10, 15]. In some works, the convolutional operator is modified to tweak the receptive field for different sized objects. [6, 18]. Hu *et al*., target the problem of face detection. Their approach consists of having two separate networks with one of them for small objects. The small object network processes an up-scaled version of the input. Additionally, they show that adding context by fusing features from multiple scales also helps. SNIP [15] uses multiple networks for different scales and train individual networks with scale specific ROIs. The outputs of each network are combined through post-processing. Our proposed changes are backbone specific; we do not add any additional parameters and we do not make any modifications to the commonly used training strategies. Additionally, we do not use multi-scale training.

Alternatively, some works proposed specialized backbones for object detection. Zhe *et al*. [17] create a modified ResNet-18 with the filter size in the first layer decreased to 3x3 and

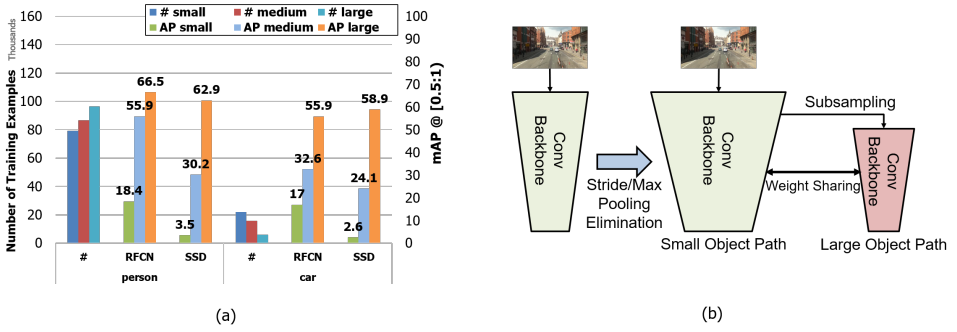

Figure 1: (a) Object detection performance for small, medium and large objects for the person and car class on the MS COCO 2017 validation set compared with the number of training samples in the MS COCO 2017 training set. The RFCN detector has a ResNet-101 backbone while the SSD detector has a Mobilenet v1 backbone. (b) Our proposed modifications to the CNN backbone. By eliminating the strided convolutions or max pooling layers, we optimize the backbone for object detection particularly detection for small objects. A dual path is added for large objects.

elimination of the max pool layer. They also train their detector from scratch with additional Batchnorm layers in the predictor. Shen *et al*. [14] create an amalgamation of DenseNet and FPN concepts and use the concatenated features for the detector. Li *et al*. [7] also develop a specialized detector for object detection with initial stride eliminated and additional layers at the termination. Contrary to the above mentioned our approach is to take existing backbones *pre-trained on ImageNet* and open up the strides. In addition, we do not add any more parameters to the base network. Instead we downsample the intermediate feature maps to recover large object performance. Our approach allows us to take advantage of ImageNet pre-training and speed up convergence while being parameter efficient.

3 Background

3.1 Analysis

An analysis of recent object detection architectures shows a wide disparity in performance between small object detection and medium/large object detection. This gap is a common element across various backbones and various architectures. We analyze the number of instances in the MS COCO training set and their corresponding performance on the MS COCO validation set with a complex and simple object detector as shown in Figure 1. Our analysis suggests that the number of training instances does not contribute to the poor performance of small object detection. We present two cases to justify this. First, for the person class, we have a balanced distribution of small, medium and large objects. However, for both RFCN and SSD there is a large gap between performance (37.5% and 26.7% respectively). For the car class, the opposite is not true. Small object detection performance is still poor (38.9% and 56.3%) even though the number of small object training instances outnumber the medium and large object training instances. This leads us to believe that the object-specific features are being impacted.

Recent works [6, 15] show ImageNet pretraining of convolutional backbones as a source of this bias. To mitigate this, He *et al*. train object detectors from scratch and achieve similar performance (compared to ImageNet pretraining); at the expense of training for more iterations [4]. However, their results shows that the performance gap for small objects still persists. We argue that the ImageNet bias is not only reflected in the pretraining but also in the way convolutional backbones are designed. It is common and accepted practice to utilize backbones designed for classification and plug them into object detection architectures such as Faster RCNN and SSD with little architectural modifications.

We conducted some preliminary experiments to show the dependence on the backbone itself. For our case study we choose a mobile object detector SSD Mobilenet v1. Mobilenetv1 [5] is a lightweight convolutional backbone developed for mobile applications. It has 1 layer of convolution and 13 layers of depth-wise separable convolutions. Within the SSD architecture, 6 predictors are used. The first 2 predictors use feature maps from the 12th and 14th layer of the backbone. 4 additional depthwise separable convolution layers are added and trained within the SSD framework. The output feature map sizes are 19x19, 10x10, 5x5, 3x3, 2x2 and 1x1. Our base implementation shows mAP of 18.8 on the MS COCO 2017 minival dataset. The small object AP is 1.8% only. Compared to the small object performance, AP medium is 19.1 and AP large is 36.3. There is almost a 30x reduction in performance for small objects.

Table 1: Evaluation on MS COCO 2017 dataset validation split for some baseline methods

Network	AP small	AR small	Params
SSDMnetv1	1.8	6.4	16.5
SSDMnetv1 Size512	6.3	14.5	16.5
SSDMnetv1 Early	2.1	8.2	17.3
SSDMnetv1 Anchor10	2.1	6.6	16.5

We tried 3 preliminary approaches. We characterized the performance of our network by increasing image size. We added predictors corresponding to feature maps earlier within the pipeline. We also added anchor boxes corresponding to small object sizes within the output of the first predictor in the base configuration. Preliminary summary results are presented in Table 1.

Large Image Size : An easy way to increase small object performance is to upsample the input image resolution. We increased the input image size from 300 x 300 to 512 x 512 and re-trained the base network. With increased compute of 4.45 GFLOPS, we got an increase in AP small by 4.5% and an increase in AR small by 8.1%.

Early Layer Prediction : As we go deeper into the pipeline, the effective stimuli of small objects decreases and contribution of neighbouring pixels increases. Adding a predictor early into the pipeline is a natural next step. We experimented with an extra predictor connnected to the 6th depthwise convolutional layer in Mobilenet v1. However, AP only improved by 0.3% while AR improved by 1.8%. This result shows that depth is an important factor that contributes to object detection accuracy and the features within the early layer are not semantically strong enough for accurate prediction.

Small Anchor Boxes : Prior information is incorporated in the object detection architectures through anchor boxes. They also allow prediction of multiple objects with overlapping regions and different aspect ratios. We added anchor boxes that corresponded to a size of 10 x 10 within the predictor working on the largest size feature map. However, increase in AP was similar to Early Layer Prediction while AR did not see any boost at all.

In the next section, we will show how our modifications to the backbone can increase small object detection performance without any changes to the training process.

4 Proposed Approach

We propose two changes to the backbone used for object detection. One, we eliminate strided convolutions and max pooling to enlarge the feature maps. Two, we use a separate path with weight sharing for detecting large objects.

4.1 Strided Convolution and Pooling Elimination

Strided convolutions and max-pooling are considered an essential part of a convolutional backbone. They serve to enhance translation invariance and also decrease the spatial dimensions of the feature maps as we go deeper into the pipeline. The effective receptive field of each neuron grows larger due to this. This works quite well in practice for tasks such as image classification but creates problems for location-sensitive tasks such as object detection. For example in SSD Mobilenet v1, the largest feature map used for prediction is of size 19 x 19 for an input image size of 300 x 300. Naively, the effective receptive field is 16 x 16 but the nature of the convolutional operator means that contribution of neighbouring features also grows to the feature map size.

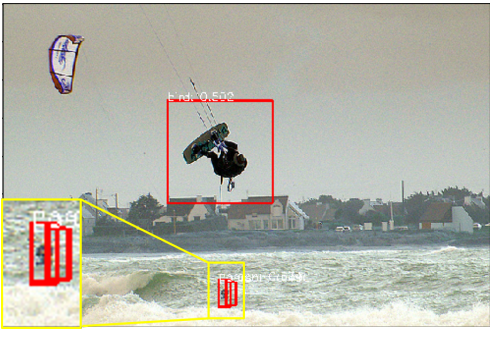

Figure 2: S2L4-6 configuration without any fine-tuning of the weights. The small person (size 15 x 19) is detected. Large object is detected but mis-classified as the network is not fine-tuned in the new configuration. Yellow Inset: Small person Detection Confidence:0.575

We created a toy experiment and eliminated strided convolutions in a few layers without fine-tuning. Surprisingly, we are able to detect the small person without any changes to the weights of the networks as shown in Figure 2. This shows that existing filter weights

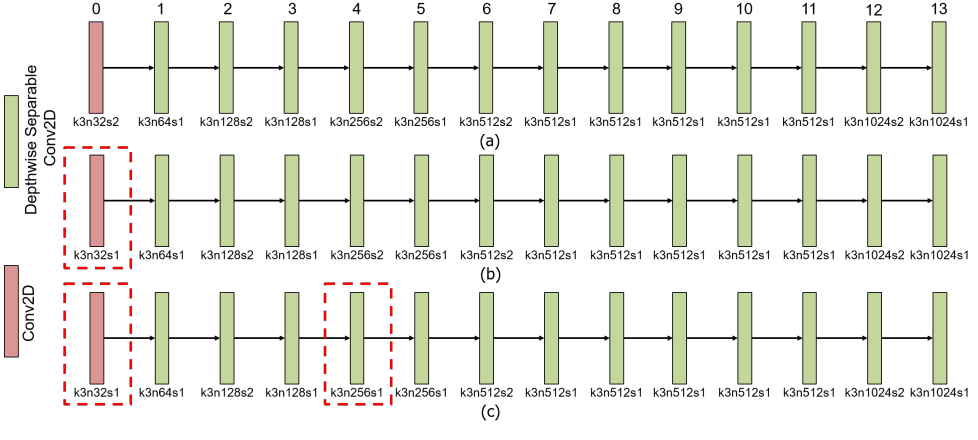

Figure 3: Stride elimination in Mobilenet v1. (a) Base Configuration. (b) S1L0 and (c) S2L0-5

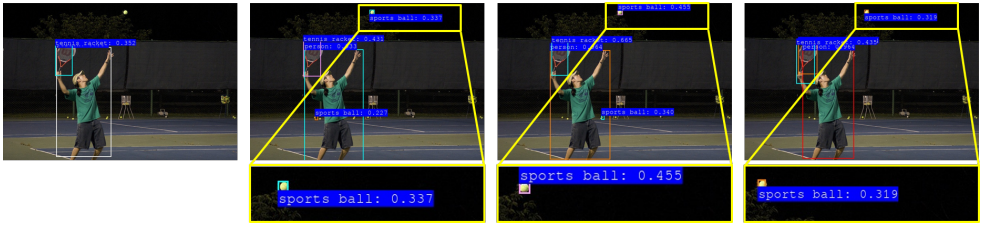

Figure 4: Example from MS COCO 2017 val. Left to right: Baseline, S1L6, S1L4, S1L0

can be used for classifying small objects. The price for this is the enlargement of the search space (by the increased size of the feature maps) and the compute required for one image. With this observation, we proceed with stride elimination within the backbone in a principled manner. We create multiple configurations of SSD mobilenet v1 with strided convolutions eliminated. A naming convention is established that allows us to easily track changes to the configuration of the base network. For each configuration the naming convention is parameterized by the number of strides eliminated S and the layer at which they are eliminated L. For example, in SSD Mobilenetv1 S1-L0 a single strided convolution is eliminated at layer 0. Two example configurations are shown in Figure 3.

We created a number of different configurations with strides eliminated at various parts of the network. In all our configurations we saw increasing small object performance but at the price of increased compute. Our most effective configurations eliminated strides early on in the pipeline as opposed to later (Section 5). We believe this is because the semantic features corresponding to smaller objects are less influenced by neighbouring distractors. In later stage stride elimination, the contribution from the small objects gets diminished earlier. At the same compute order, early stage stride elimination shows the best results. A visual comparison is shown from MS COCO val 2017 in Figure 4. In all of our modified configurations, the small tennis ball is correctly detected while in the baseline it is not.

4.2 Dual-Path Weight Sharing for Large Object Detection

The side effect of stride elimination is reduced performance on large objects. Due to the decreased receptive field of the neurons in the backbone, the features for large objects are not aggregated effectively. Past approaches [6, 15] train separate networks with regular sized inputs for large objects and up-scaled inputs for small objects. The separate networks have similar structures but different weights. We propose an alternative weight-sharing scheme for recovering large object performance. Let our small object detector be parameterized by S and L. For every l in L, the output feature maps are larger inversely proportional to the reduced stride. For large objects, the enlarged feature map is subsampled and then propagated through the same networks. In this way, a dual path is created but without adding any more parameters.

There are two ways to accommodate the additional path for prediction. In our concatenation configuration, we concatenate the scale-specific feature maps according to their respective layers i.e. for S1L0, the feature map for the 13th layer is 19 x 19 and in the base configuration the 11th feature map is of size 19 x 19. Thus, the 13th feature map of the small object path and the 11th feature map of the large object path are concatenated and provided to the prediction layers. The process is shown in Figure 5. This modification when applied to

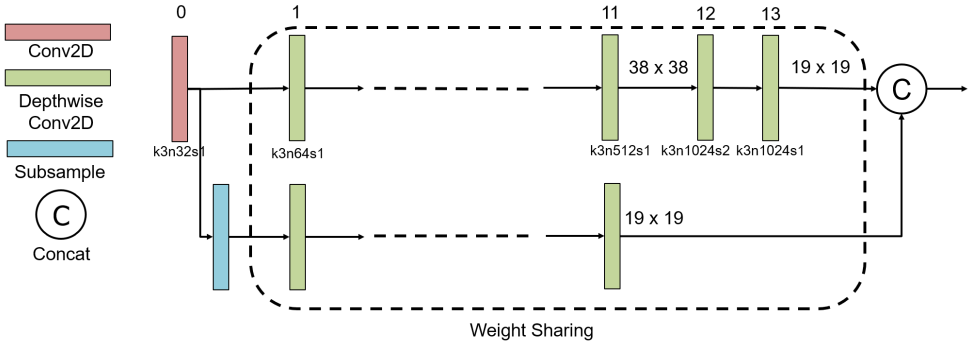

Figure 5: Dual path for large and small objects in S1L0. The separate path for large objects is created by subsampling the enlarged feature maps. The weights are shared between the two paths

S1L0 recovers AR large and AP large closer to their base values. However, this modification increases the number of parameters and compute. We propose another alternative where the predictors for small objects and large objects are separated. In the case of SSD Mobilenet v1 S1L0, the 1st and 2nd predictors work only on the small object path feature maps (38 x 38 and 19 x 19) while the predictors 3 to 6 work only on the large object path feature maps (10 x 10 onwards). With this approach, our dual path has no additional parameters. We term these configurations with the *C* and *S* suffix respectively.

5 Experimental Results

We present the results of our proposed changes on SSD Mobilenet v1. For training, we do not change the training schedule for any of our networks. The detector is trained using SGD for a maximum of 100 epochs using a cosine learning rate decay schedule. The base learning rate is set to 0.001. The batch size is set to 32 and is trained on two GTX 1080 Ti GPUs using asynchronous SGD. The backbone layers are pre-trained on ImageNet while the extra layers for prediction are trained from scratch. The prior boxes for each configuration are adjusted depending on the size of the feature maps. For example, in all S1 configurations, the largest feature map increases in size from 19 x 19 to 38 x 38. The size of the base prior box for that layer is then decreased by a factor of 2 from 16 to 8. All experiments were conducted using Pytorch 1.0[1]. The summary results of our experiments are presented in Table 2.

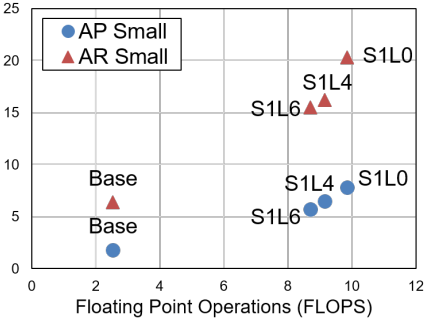

Figure 6: Early vs. Late Stride elimination and its effect on Average Precision (AP) and Average Recall (AR) versus the compute complexity added.

Stride Elimination: Early or Later? We determine whether it is beneficial to eliminate

[1]Code is available at https://github.com/burhanmudassar/smallObject.git

Table 2: Evaluation on the MS COCO 2017 dataset val split with proposed configurations

Network	AP	AP small	AP medium	AP large	AR small	AR medium	AR large	Compute (GFLOPS)	Params
Mobile-friendly Backbone									
SSDMnetv1	18.8	1.8	19.1	36.3	6.4	36.7	58.5	2.5	16.5
SSDMnetv1 S1L6	18.6	5.7	22.6	28.3	15.5	41.6	51.3	8.7	16.5
SSDMnetv1 S1L4	18.3	6.5	23.1	27.6	16.2	41.9	51.3	9.2	16.5
SSDMnetv1 S1L0	19.6	7.8	25.2	27.7	20.3	46.8	51.6	9.9	16.5
SSDMnetv1 S1L4-Fmap1	19.2	6.0	23.5	28.6	19.4	45.1	52.6	9.2	16.5
SSDMnetv1 S1L0-Fmap1	19.6	7.9	25.2	28.6	20.5	46.7	53.3	9.9	16.5
SSDMnetv1 S1L0-DualPath19-C	19.8	7.7	25.1	29.8	19.9	46.8	53.4	11.9	19.2
SSDMnetv1 S1L0-DualPath19-10-C	20.7	7.7	26.4	30.1	20.8	47.5	55.2	12.3	23.9
SSDMnetv1 S1L0-DualPath19-10-S	20.7	8.2	25.0	30.8	21.2	46.3	54.4	10.5	16.5
Complex Backbone									
SSD VGG-16	24.6	6.9	27.9	40.3	16.0	51.3	63.8	35.6	35.1
SSD VGG-16 S1L2	23.7	11.2	27.3	33.4	28.0	50.5	57.1	128.9	35.1
SSD VGG-16 S1L2-DualPath	28.5	13.7	31.4	41.3	30.4	53.9	63.4	154.1	35.1

Figure 7: Comparison on MS COCO 2017 val. Top: Baseline. Bottom: S1L0. A detection threshold of 0.2 is used for both cases. Using our approach we are able to detect small objects of various categories in a wide variety of scenes.

the stride early in the pipeline or later. We fix S to be 1 and experiment with different values of L and evaluate small object performance. Our results indicate that stride elimination early in the pipeline leads to the largest increase in AP small. Moreover, in terms of compute the difference between our L6 and L0 is only 1.14 GFLOPS. In terms of mAP, L0 shows the largest increase of AP small over the base network (6.0%). Between L6 and L0, the AP small increases by 2.1%. In terms of Recall, we are able to increase the AR small by 13.9% over the base configuration with S1L0. Not only AP/AR small but AP/AR medium also improves by 6.1/10.1%. The results of single stride elimination are summarized in Figure 6. With a single stride eliminated the smallest feature map enlarges from 2 x 2 to 1 x 1. As one possible reason for reduced performance on large objects, we increase the stride of the last extra layer in SSD Mobilenet v1. This reduces the feature map size from 2 x 2 to 1 x 1. For S1L4-Fmap1 and S1L0-Fmap1 the AR large increases by 1.3% to 1.7% and the AP large increases by 1.0% to 0.9%. Some example comparisons between S1L0 and the baseline are presented in Figure 7.

Per-Class Improvement: Investigating the per-class performance, we observe that improvement varies per class. For the S1L0 configuration the person class AP improves from 3.5% to 11.4% and the Car AP improves from 2.6% to 10.2%.

Multi-Stride Elimination : We determine the benefit of eliminating more than 1 stride within the network. For S2L4-6, the compute increases by 30.9 GFLOPs and the increase in AP small is marginal compared to the amount of compute added. For example, between

Table 3: Comparison with prior work on MS COCO val 2017.

Network	AP	AP small	AP medium	AP large	AR small	AR medium	AR large	Compute (GFLOPS)	Params (Millions)
Mobile-friendly Backbone									
RFB MobileNet 300 [11]	20.7	1.7	21.0	38.3	4.8	35.1	55.9	1.6	7.7
PeleeNet Mobilenet [16]	21.8	3.4	20.9	**40.0**	7.2	34.5	**56.2**	1.3	5.4
SSDMnetv1 S1L0-DualPath-S	20.7	**8.2**	**25.0**	30.8	**21.2**	**46.3**	54.4	10.5	16.5
Complex Backbone									
RFB-Net 300 [11]	30.2	11.9	**33.0**	**47.8**	19.1	47.5	**63.6**	39.9	45.1
SSD VGG-16 S1L2-DualPath	28.5	**13.7**	31.4	41.3	**30.4**	**53.9**	63.4	154.1	35.1

Table 4: Measured Runtimes on a GPU (GTX 1080 Ti)

	SSD Mobilenet v1						SSD VGG-16	
Network	Base	Size512	S1L6	S1L4	S1L0	DualPath19-10-S	Base	S1L2
Runtime (ms)	6.9	6.9	6.9	7.8	10.4	12.1	12.1	31.5

S1L6 and S2L4-6, the AP small increase is only 0.7%.

Dual-Path : For our dual-path experiments, we choose the S1L0 configuration. We create two configurations for concatenation. In the first, the 11th feature map for the large object path (19 x 19) is concatenated with the 13th feature map of the small object path (Dualpath-19-C). In the second configuration, the 13th feature map of the large object path (10 x 10) is also concatenated with the 14th feature map of the small object path (Dualpath-19-10-C). For the dual-path configurations, compute increases by 2.0 GFLOPS and 2.48 GFLOPS respectively while parameters increase by 2.6 and 7.3 Million Params respectively. The parameter increase comes from the increase in channel dimensions for the two predictors. The dual-path configuration recovers the AP/AR large by 1.5/1.9% respectively. The split dual path (Dualpath-19-10-S) removes the additional parameter overhead and maintains the AP/AR large recovery.

Extension to Complex Backbones : We implement our ideas on a significantly more complex backbone VGG16. There are no strided convolutions in VGG but there are max pooling layers. We eliminate the max pool layer in layer 3. To keep the feature size same at the last layers, we increase the kernel size and stride of the max pooling layers at the later stage of the network. In this way, the network and training configuration does not need to be changed. The S1L2 Dualpath configuration improves AP/AR small by 6.8/14.4% respectively.

Comparison with Prior Work: Compared to RFB Net [11], our SSD-VGG network shows an increase in AP/AR small by 1.8% and 11.3% respectively. Our mobilenet configuration shows an AP/AR small increase of 6.5% and 16.4% respectively. Compared to PeleeNet [16], our network shows an AP/AR small increase of 4.8% and 14% respectively. The comparison is shown in Table 3.

Measured Latency : We evaluate the latency of the detectors on a GTX 1080 Ti Platform by evaluating the runtime for a single image inference for a 1000 runs. For single image inference, we add 5.2 ms of latency for S1L0 as shown in Table 4. The measured time does not include preprocessing (Resizing/Mean Subtraction) or postprocessing (NMS) as they are common across all the architectures.

6 Conclusion

In this work, we have shown how simple modifications to the convolutional backbone for object detectors can increase small object performance. Our approach involves enlargement of the intermediate feature maps by eliminating strided convolutions and max pool operators. Our approach increases precision and recall for small objects by a wide margin. However, the enlargement of the feature maps reduces the performance on large objects. We propose a dual-path weight sharing scheme to recover that. Our approach does not add any more parameters to the network but increases the required compute showing that our networks are more parameter efficient. Additionally, all the benefits of ImageNet pretraining can be used as it is trivial to change the stride operation and pooling operator without changing the weights. For future work, we intend to work on improving the classification accuracy. Our current results show that the localization accuracy (AR) improves by a much wider margin than classification accuracy (AP).

Acknowledgement The research reported here was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-17-2-0045. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA.

References

[1] Fatal arizona crash: Uber car saw woman; called it a false positive. https://www.extremetech.com/extreme/268915-fatal-arizona-crash-ubercar-saw-woman-called-it-a-false-positive. Accessed: 2019-07-12.

[2] 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*, pages 379–387, 2016.

[3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

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

[5] 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.

[6] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017.

[7] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: A backbone network for object detection. *arXiv preprint arXiv:1804.06215*, 2018.

[8] Zuoxin Li and Fuqiang Zhou. Fssd: feature fusion single shot multibox detector. *arXiv preprint arXiv:1712.00960*, 2017.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[11] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 385–400, 2018.

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

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[14] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1919–1927, 2017.

[15] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3578–3587, 2018.

[16] Robert J Wang, Xiang Li, and Charles X Ling. Pelee: A real-time object detection system on mobile devices. In *Advances in Neural Information Processing Systems*, pages 1963–1972, 2018.

[17] Rui Zhu, Shifeng Zhang, Xiaobo Wang, Longyin Wen, Hailin Shi, Liefeng Bo, and Tao Mei. Scratchdet: Training single-shot object detectors from scratch. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[18] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. *arXiv preprint arXiv:1811.11168*, 2018.