# Light-Weight RefineNet for Real-Time Semantic Segmentation

Vladimir Nekrasov
vladimir.nekrasov@adelaide.edu.au
Chunhua Shen
chunhua.shen@adelaide.edu.au
Ian Reid
ian.reid@adelaide.edu.au

School of Computer Science,
The University of Adelaide,
Australia

**Abstract**

We consider an important task of effective and efficient semantic image segmentation. In particular, we adapt a powerful semantic segmentation architecture, called *RefineNet* [46], into the more compact one, suitable even for tasks requiring real-time performance on high-resolution inputs. To this end, we identify computationally expensive blocks in the original setup, and propose two modifications aimed to decrease the number of parameters and floating point operations. By doing that, we achieve more than twofold model reduction, while keeping the performance levels almost intact. Our fastest model undergoes a significant speed-up boost from 20 FPS to 55 FPS on a generic GPU card on $512 \times 512$ inputs with solid 81.1% mean iou performance on the test set of PASCAL VOC [18], while our slowest model with 32 FPS (from original 17 FPS) shows 82.7% mean iou on the same dataset. Alternatively, we showcase that our approach is easily mixable with light-weight classification networks: we attain 79.2% mean iou on PASCAL VOC using a model that contains only 3.3M parameters and performs only 9.3B floating point operations.

## 1 Introduction

The number of empirical successes of deep learning keeps steadily increasing, and new technological and architectural breakthroughs become available practically every month. In particular, deep learning has become the default choice in most areas of computer vision, natural language processing, robotics, audio processing [21, 22, 24, 37, 40, 44, 57, 61, 65, 67, 69, 70, 72, 75]. Nevertheless, these breakthroughs often come at a price of expensive computational requirements, which hinders their direct applicability in tasks requiring real-time processing. The good part of this story is that it has been empirically shown by multiple researchers that oftentimes there are lots of redundant operations [1, 16, 17, 27, 31, 36, 68], and this redundancy can be (and should be) exploited in order to gain speed benefits while keeping performance mostly intact. The bad part of the story is that there are not (yet) general approaches on how to exploit such redundancies in most cases. For example, such methods as knowledge distillation [1, 3, 31, 58] and pruning [26, 27, 29, 42] require us to have an access to an already pre-trained large model in order to train a smaller model of the same (or near) performance. Obviously, there might be times when this is not feasible. In contrast, designing a novel architecture for a specific scenario of high-resolution inputs [54, 76] limits

the applicability of the same architecture on datasets with drastically different properties and often requires expensive training from scratch.

Here we are restricting ourselves to the task of semantic segmentation, which has proven to be pivotal on the way of solving scene understanding, and has been successfully exploited in multiple real-world applications, such as medical image segmentation [11, 59], road scene understanding [2, 70], aerial segmentation [38, 51]. These applications tend to rely on real-time processing with high-resolution inputs, which is the Achilles' heel of most modern semantic segmentation networks.

Motivated by the aforementioned observations we aim to tackle the task of real-time semantic segmentation in a different way. In particular, we build our approach upon *RefineNet* [46], a powerful semantic segmentation architecture that can be seamlessly used with any backbone network, such as ResNet [30], DenseNet [33], NASNet [82], or any other. This architecture belongs to the family of the 'encoder-decoder' segmentation networks [2, 53, 54], where the input image is first progressively downsampled, and later progressively upsampled in order to recover the original input size. We have chosen this particular architecture as it i) shows the best performance among the 'encoder-decoder' approaches, and ii) does not operate over large feature maps in the last layers as methods exploiting atrous convolution do [4, 5, 7, 74, 77]. The processing of large feature maps significantly hinders real-time performance of such architectures as DeepLab [4, 5, 7] and PSPNet [77] due to the increase in the number of floating point operations. As a weak point, the 'encoder-decoder' approaches tend to have a larger number of parameters due to the decoder part that recovers the high resolution output. In this work, we tackle the real-time performance problem by specifically concentrating on the decoder and empirically showing that both the number of parameters and floating point operations can be drastically reduced without a significant drop in accuracy.

In particular, i) we outline important engineering ameliorations that save the number of parameters by more than 50% in the original RefineNet; then ii) we pinpoint the redundancy of residual blocks in the architecture and show that the performance stays intact with those blocks being removed. We conduct extensive experiments on three competitive benchmarks for semantic segmentation - PASCAL VOC [18], NYUDv2 [64] and PASCAL Person-Part [6, 9], and with five different backbone networks - ResNet-50, ResNet-101, ResNet-152 [30], along with recently released NASNet-Mobile [82], and MobileNet-v2 [60]. The first three backbones are used for the direct comparison between our approach and the original RefineNet, while the last two are used to showcase that our method is orthogonal to the backbone compression, and that we can further benefit from combining these methods.

Quantitatively, our fastest ResNet model achieves 55 FPS on inputs of size $512 \times 512$ on GTX 1080Ti, while having 81.1% mean intersection over union (mean iou) performance on PASCAL VOC, 41.7% mean iou on NYUDv2, and 64.9% mean iou on Person-Part. In contrast, our best performing model with 82.7% mean iou on VOC, 44.4% mean iou on NYUDv2, and 67.6% mean iou on Person-Part, still maintains solid 32 FPS.

All the models will be made publicly available to facilitate the usage of effective and efficient semantic segmentation in a multitude of applications.

## 2 Related work

**Semantic segmentation.** Early approaches in semantic segmentation relied on handcrafted features, such as HOG [14] and SIFT [50], in combination with plain classifiers [13, 20, 62, 63] and hierarchical graphical models [39, 41, 56]. This lasted until the resurgence of

deep learning, and a pivotal work by Long *et al*. [49], in which the authors converted a deep image classification network into a fully convolutional one, able to operate on inputs of different sizes. Further, this line of work was extended by using dilated convolutions and probabilistic graphical models [4, 45, 48, 74, 78]. Other works concentrated around the encoder-decoder paradigm [46, 53, 54], where the image is first progressively downsampled, and then progressively upsampled in order to generate the segmentation mask of the same size as the input. Most recently, Lin *et al*. [46], proposed RefineNet that extended the encoder-decoder approach by adding residual units inside the skip-connections between encoder and decoder. Zhao *et al*. [77] appended a multi-scale pooling layer to improve the information flow between local and global (contextual) information; Fu *et al*. [19] exploited multiple encoder-decoder blocks arranged in an hour glass manner [73] and achieved competitive results using DenseNet [33]. The current state-of-the art on the popular benchmark dataset PASCAL VOC [18] belongs to DeepLab-v3 [7] (86.9% mean intersection-over-union on the test set) and DeepLab-v3+ [8] (89.0%) based on ResNet [30] and Xception [10], correspondingly, where the authors included batch normalisation [35] layers inside the atrous spatial pyramid pooling block (ASPP) [5], made use of a large multi-label dataset [66], and added decoder in the latest version.

**Real-Time Segmentation.** Most methods outlined above either suffer from a large number of parameters [46], a large number of floating point operations [4, 5, 7, 77], or both [49, 53, 54, 73]. These issues constitute a significant drawback of exploiting such models in applications requiring real-time processing.

To alleviate them, there have been several task-specific approaches, e.g., ICNet [76], where the authors adapt PSPNet [77] to deal with multiple image scales progressively. They attain the speed of 30 FPS on $1024 \times 2048$ images, and 67% mean iou on the validation set of CityScapes [12], but it is not clear whether this approach would still acquire solid performance on other datasets with low-resolution inputs. Alternatively, Li *et al*. [43] propose a cascading approach where in each progressive level of the cascade only a certain portion of pixels is being processed by explicitly setting a hard threshold on intermediate classifier outputs. They demonstrate performance of 14.3 FPS on $512 \times 512$ input resolution with 78.2% mean iou on PASCAL VOC [18]. We also note that several other real-time segmentation networks have been following the encoder-decoder paradigm, but have not been able to acquire decent performance levels. Concretely, SegNet [2] achieved 40 FPS on inputs of size $360 \times 480$ with only 57.0% mean iou on CityScapes, while ENet [54] were able to perform inference of 20 FPS on inputs of size $1920 \times 1080$ with 58.3% mean iou on CityScapes.

**Real-Time Inference in Other Domains.** Multiple task-agnostic solutions have been proposed to speed-up inference in neural networks by compression. Current methods achieve more than $100\times$ compression rates [34]. Among them, most popular approaches include quantisation [23, 79, 81], pruning [26, 27, 29, 42] and knowledge distillation [1, 3, 31, 58]. Other notable examples are centered around the idea of low-rank factorisation and decomposition [17, 36], where a large layer can be decomposed into smaller ones by exploiting (linear) structure.

While all the methods above have proven to be effective in achieving significant compression rates, an initial powerful, but large pre-trained model for the task at hand must be acquired first. We further note that these methods can be applicable as a post-processing step on top of our approach, but we leave this direction for later exploration.

Finally, most recently, a multitude of new, light-weight architectures [32, 34] has been

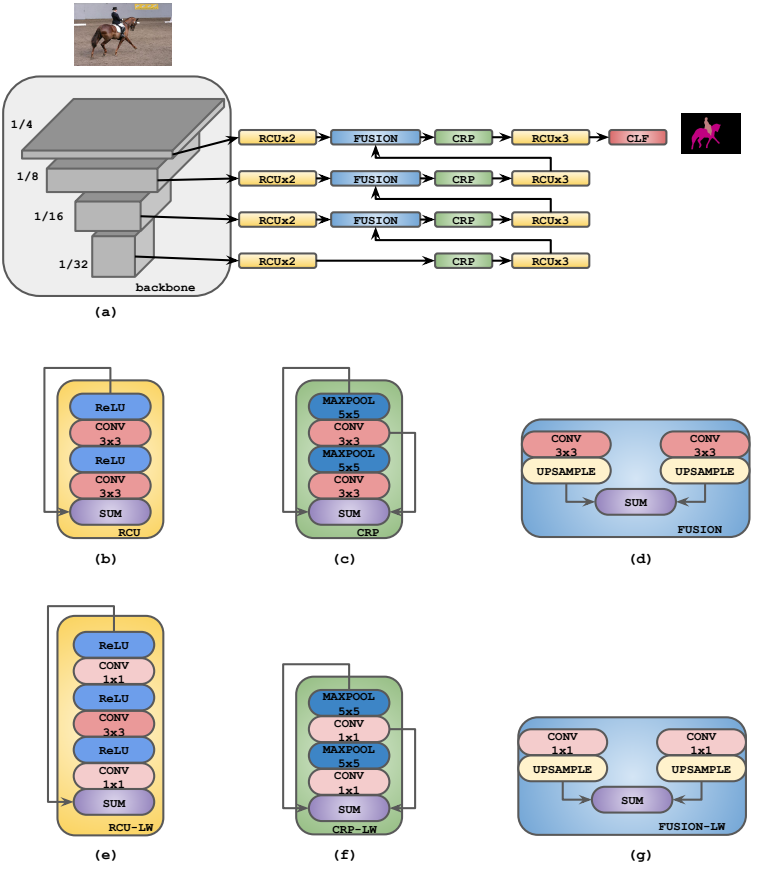


Figure 1: RefineNet structure. (a) General network architecture with RefineNet for semantic segmentation, where *CLF* stands for a single 3x3 convolutional layer with the number of channels being equal to the number of output classes; (b)-(d) general outline of original RCU, CRP and fusion blocks; (e)-(g) light-weight RCU, CRP and fusion blocks. In the interests of brevity, we only visualise 2 convolutional layers for the CRP blocks (instead of 4 used in the original architecture). Note that we do not use any RCU blocks in our final architecture as discussed in Sec. 3.3.

proposed, and it has empirically been shown that models with a smaller number of parameters are able to attain solid performance. To highlight that our approach is orthogonal to such advances in the design of classification networks, we conduct experiments with two light-weight architectures, namely, NASNet-Mobile [82], and MobileNet-v2 [60], and show competitive results with very few parameters and floating point operations.

# 3 Light-Weight RefineNet

Here we will outline our approach aimed to decrease the number of parameters and floating point operations of the original RefineNet architecture, while keeping the performance levels the same.

We start by describing the original setup, and then continue with the explanation and discussion of the changes that we propose to be made.

### 3.1 RefineNet Primer

As mentioned above, the RefineNet architecture belongs to the family of the encoder-decoder approaches. As the encoder backbone, it can re-use most popular classification networks; in particular, the original contribution was showcased using residual networks [30]. The approach does not amend an underlying classification network in any way except for omitting last classification layers.

In the decoder part, RefineNet relies on two types of additional abstractions: residual convolutional unit (RCU) (Fig. 1(b)) and chained residual pooling (CRP) (Fig. 1(c)). The first one is a simplification of the original residual block [30] without batch normalisation [35], while the second one is a sequence of multiple convolutional and pooling layers, also arranged in a residual manner. All of these blocks use $3 \times 3$ convolutions and $5 \times 5$ pooling with appropriate padding so that the spatial size would stay intact.

The decoding process starts with the propagation of the last output from the classifier (with the lowest resolution) through two RCU blocks followed by four pooling blocks of CRP and another three RCU blocks before being fed into a fusion block (Fig. 1(d)) along with the second to last feature map. Inside the fusion block, each path is convolved with $3 \times 3$ convolution and upsampled to the largest resolution among the paths. Two paths are then summed up, and analagoulosly further propagated through several RCU, CRP and fusion blocks until the desired resolution is reached. The final convolutional layer produces the score map.

### 3.2 Replacing 3x3 convolutions

We start by highlighting that the most expensive parts in terms of both the number of parameters and the number of floating point operations of the original RefineNet stem from the ubiquitous usage of $3 \times 3$ convolutions. Thus, we focus on replacing them with simpler counterparts without performance drops.

Intuitively, convolutions with larger kernel sizes aim to increase the receptive field size (and the global context coverage), while $1 \times 1$ convolutions are merely transforming per-pixel features from one space to another locally. We argue that in the case of RefineNet we do not require expensive $3 \times 3$ convolutions at all, and we are able to show empirically that replacing them with $1 \times 1$ convolutions does not hurt performance. Furthermore, we evaluate the empirical receptive field size [80] for both cases, and do not find any significant difference (Section 5.1). In particular, we replace $3 \times 3$ convolutions within CRP and fusion blocks with $1 \times 1$ counterpart (Fig. 1(f-g)), and we amend RCU into the one with the bottleneck design [30] (Fig. 1(e)). This way, we reduce the number of parameters by more than $2\times$ times, and the number of FLOPs by more than $3\times$ times (Table 1).

Alternatively, we might have reverted to recently proposed depthwise separable convolutions [10] that are proven to save lots of parameters without sacrificing performance, but our approach is even simpler as it does not use any $3 \times 3$ convolutions at all, except for the last classification layer.

### 3.3 Omitting RCU blocks

We proceeded with initial experiments and trained the light weight architecture outlined in the previous section using PASCAL VOC [18]. We were able to achieve close to the original network performance, and, furthermore, observed that removing RCU blocks did not lead

| Model | Parameters,M | FLOPs,B |
|---|---|---|
| RefineNet-101 [46] | 118 | 263 |
| **RefineNet-101-LW-WITH-RCU** | 54 | 76 |
| **RefineNet-101-LW** | **46** | **52** |

Table 1: Comparison of the number of parameters and floating point operations on $512 \times 512$ inputs between original RefineNet, Light-Weight RefineNet with RCU (**LW-WITH-RCU**), and Light-Weight RefineNet without RCU (**LW**). All the networks exploit ResNet-101 as backbone.

to any accuracy deterioration, and, in fact, the weights in RCU blocks almost completely saturated.

To confirm that this only occurs in the light weight case, we explored dropping RCU blocks in the original RefineNet architecture, and we experienced more than 5% performance drop on the same dataset. We argue that this happens due to the RCU blocks being redundant in the $1 \times 1$ convolution regime, as the only important goal of increasing the contextual coverage is essentially performed by pooling layers inside CRP. To back up this claim, we conduct a series of ablation experiments which are covered later in Section 5.2.

Our final architecture does not contain any RCU blocks and only relies on CRP blocks with $1 \times 1$ convolutions and $5 \times 5$ max-pooling inside, which makes our method extremely fast and light-weight.

### 3.4 Adaptation to different backbones

A significant practical benefit of the RefineNet architecture is that it can be mixed with any backbone network as long as the backbone has several subsampling operations inside (which is the case for most SOTA classifiers). Thus, there are no specific changes needed to be made in order to apply the RefineNet architecture using any other model. In particular, in the experiments section, we showcase its adaptation using efficient NASNet-Mobile [82] and MobileNet-v2 [60] networks; we still achieve solid performance with the limited number of parameters and floating point operations.

## 4 Experiments

In the experimental part, our aims are to prove empirically that we are able i) to achieve similar performance levels with the original RefineNet while ii) significantly reducing the number of parameters and iii) drastically increasing the speed of a forward pass; and iv) to highlight the possibility of applying our method using other architectures.

To this end, we consider three segmentation datasets, namely, NYUDv2 [64], PASCAL VOC [18] and PASCAL Person-Part dataset [6, 9], and five classification networks, i.e., ResNet-50, ResNet-101, ResNet-152 [30], NASNet-Mobile [82] and MobileNet-v2 [60] (only for Pascal VOC), all of which have been pre-trained on ImageNet [15]. As a general practice, we report mean intersection over union [18] on each benchmark. Additional results on PASCAL Context [52] and CityScapes [12] are given in the supplementary material.

We perform all our experiments in PyTorch [55], and train using stochastic gradient descent with momentum. For all residual networks we start with the initial learning rate of 5e-4, for NASNet-mobile and MobileNet-v2 we start with the learning rate of 1e-3. We keep batch norm statistics frozen during the training.

| Model | NYUD mIoU,% | Person mIoU,% | Params,M | Runtime,ms |
|---|---|---|---|---|
| FCN16-s RGB-HHA [29] | 34.0 | - | - | - |
| Context [45] | 40.6 | - | - | - |
| DeepLab-v2-CRF [5] | - | 64.9 (*msc*) | **44** | - |
| RefineNet-50 [46] | 42.5 | 65.7 | 99 | $54.18 \pm 0.46$ |
| RefineNet-101 [46] | 43.6 | 67.6 | 118 | $60.25 \pm 0.53$ |
| RefineNet-152 [46] | **46.5** (*msc*) | **68.8** (*msc*) | 134 | $69.37 \pm 0.78$ |
| **RefineNet-LW-50** (ours) | 41.7 | 64.9 | **27** | **19.56** $\pm 0.29$ |
| **RefineNet-LW-101** (ours) | 43.6 | 66.7 | **46** | **27.16** $\pm 0.19$ |
| **RefineNet-LW-152** (ours) | 44.4 | 67.6 | **62** | **35.82** $\pm 0.23$ |

Table 2: Quantitative results on the test sets of NYUDv2 and PASCAL Person-Part. Mean iou, the number of parameters and the runtime (mean$\pm$std) of one forward pass on $625 \times 468$ inputs are reported, where possible. Multi-scale evaluation is defined as *msc*.

For benchmarking, we use a workstation with 8GB RAM, Intel i5-7600 processor, and one GT1080Ti GPU card under CUDA 9.0 and CuDNN 7.0. For fair comparison of runtime, we re-implement original RefineNet in PyTorch. We compute 100 forward passes with random inputs, and average the results; when reporting, we provide both the mean and standard deviation values.

**NYUDv2.** We first conduct a series of initial experiments on NYUDv2 dataset [25, 34]. This dataset comprises 1449 RGB-D images with 40 segmented class labels, of which 795 are used for training and 654 for testing, respectively. We do not make use of depth information in any way. We reduce the learning rate by half after 100 and 200 epochs, and keep training until 300 epochs, or until earlier convergence.

Our quantitative results are provided in Table 2 along with the results from the original RefineNet, and other competitive methods on this dataset. We note that for all ResNet networks we are able to closely match the performance of the original RefineNet, while having only a slight portion of the original parameters. This further leads to a significant twofold speedup.

**Person-Part.** PASCAL Person-Part dataset [6, 9] consists of 1716 training and 1817 validation images with 6 semantic classes, including head, torso, and upper/lower legs/arms, plus background. We follow the same training strategy as for NYUDv2, and provide our results in Table 2. Again, we achieve analagous to the original models results.

**PASCAL VOC.** We continue our experiments with a standard benchmark dataset for semantic segmentation, PASCAL VOC [18]. This dataset consists of 4369 images with 20 semantic classes plus background, of which 1464 constitute the training set, 1449 - validation, and 1456 - test, respectively. As commonly done, we augment it with additionally annotated VOC images from BSD [28], as well as images from MS COCO [47].

We train all the networks using the same strategy, but with different learning rates as outlined above: in particular, we reduce the learning rate by half after 20 epochs of COCO training and after 50 epochs of the BSD training, and keep training until 200 total epochs, or until the accuracy on the validation set stops improving.

Our quantitative results on validation and test sets are given in Table 3 along with the results from the original RefineNet. We again notice that for all ResNet networks, we achieve performance on-par with the original RefineNet; for NASNet-Mobile and MobileNet-v2, we

| Model | val mIoU,% | test mIoU,% | FLOPS,B |
|---|---|---|---|
| DeepLab-v2-ResNet-101-CRF [5] | 77.7 | 79.7 | - |
| RefineNet-101 [46] | - | 82.4 | 263 |
| RefineNet-152 [46] | - | **83.4** | 283 |
| **RefineNet-LW-50** (ours) | 78.5 | 81.1[1] | **33** |
| **RefineNet-LW-101** (ours) | 80.3 | 82.0[2] | 52 |
| **RefineNet-LW-152** (ours) | 82.1 | 82.7[3] | 71 |
| MobileNet-v1-DeepLab-v3 [60] | 75.3 | - | 14.2 |
| MobileNet-v2-DeepLab-v3 [60] | 75.7 | - | **5.8** |
| **RefineNet-LW-MobileNet-v2** (ours) | **76.2** | **79.2**[4] | **9.3** |
| **RefineNet-LW-NASNet-Mobile** (ours) | **77.4** | **79.3**[5] | 11.4 |

Table 3: Quantitative results on PASCAL VOC. Mean iou and the number of FLOPs on $512 \times 512$ inputs are reported, where possible.

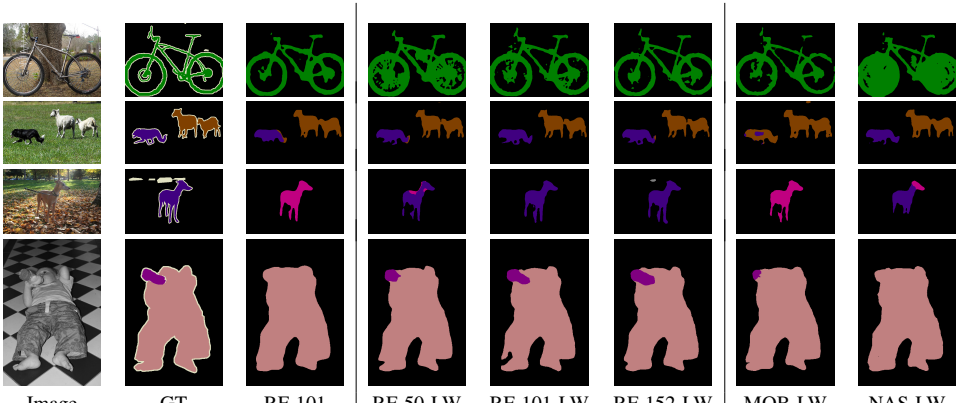


Figure 2: Visual results on validation set of PASCAL VOC with residual models (RF), MobileNet-v2 (MOB) and NASNet-Mobile (NAS). The original RefineNet-101 (RF-101) is our re-implementation.

outperform both MobileNet-v1+DeepLab-v3 and MobileNet-v2+DeepLab-v3 approaches [60], which are closely related to our method. Qualitative results are provided on Figure 2.

## 5 Discussion

As noted above, we are able to achieve similar results by omitting more than half of the original parameters. This raises the question on how and why this does happen. We conduct a series of experiments aimed to provide some insights into this matter. More details are provided in the supplementary material.

[1] http://host.robots.ox.ac.uk:8080/anonymous/ZAC8JH.html
[2] http://host.robots.ox.ac.uk:8080/anonymous/RBAJXC.html
[3] http://host.robots.ox.ac.uk:8080/anonymous/EFA17T.html
[4] http://host.robots.ox.ac.uk:8080/anonymous/90TQVN.html
[5] http://host.robots.ox.ac.uk:8080/anonymous/WRYPBD.html

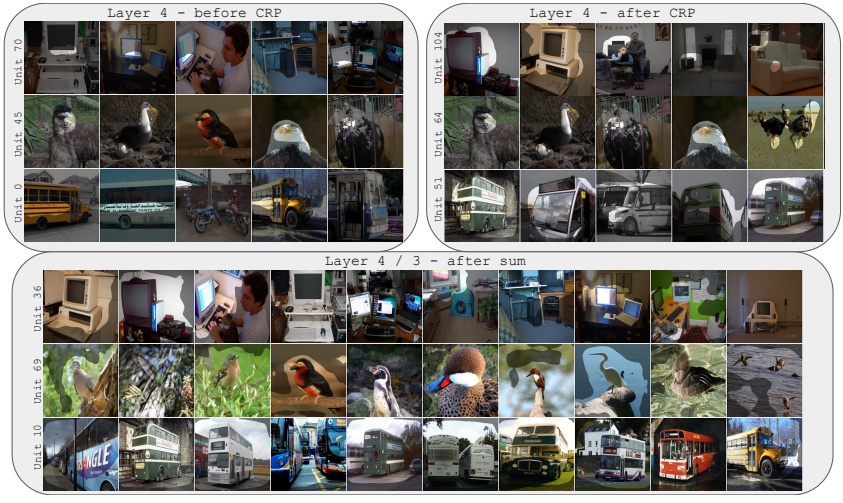


Figure 3: Comparison of empirical receptive field before CRP (top left), after CRP (top right), and after the first summation of levels 4 and 3 (bottom) in RefineNet-LW-101 pre-trained on PASCAL VOC. Top activated images along with top activated regions are shown.

## 5.1 Receptive field size

First, we consider the issue of the receptive field size. Intuitively, dropping $3 \times 3$ convolutions should significantly harm the receptive field size of the original architecture. Nevertheless, we note that we do not experience this due to i) the skip-design structure of RefineNet, where low-level features are being summed up with the high-level ones, and ii) keeping CRP blocks that are responsible for gathering contextual information.

In particular, we explore the empirical receptive field (ERF) size [30] in Light-Weight RefineNet-101 pre-trained on PASCAL VOC. Before CRP ERF is concentrated around small object parts, and barely covers the objects (Fig. 3). The CRP block tends to enlarge ERF, while the summation with the lower layer features further produces significantly larger activation contours.

## 5.2 Representational power

Even though the RCU block seems not to influence the receptive field size of the network, it might still be responsible for producing important features for both segmentation and classification.

To evaluate whether it is the case indeed, we conduct ablation experiments by adding multi-label classification and segmentation heads in the lowest resolution block of RefineNet-101 pre-trained on PASCAL VOC 1) before RCU, 2) after RCU, and 3) after CRP. We fix the rest of the trained network and fine-tune only these heads separately using 1464 training images of the VOC dataset. We evaluate all the results on the validation subset of VOC.

As seen from Table 4, CRP is the main driving force behind accurate segmentation and classification, while the RCU block improves the results just marginally.

| Model | mIoU,% | acc,% |
|---|---|---|
| Before RCU | 54.63 | 95.39 |
| After RCU | 55.16 | 95.42 |
| After CRP | **57.83** | **97.71** |

Table 4: Ablation experiments comparing CRP and RCU. Multi-label accuracy (without background) and mean IoU are reported as measured on the validation set of PASCAL VOC.

# 6 Conclusions

In this work, we tackled the problem of rethinking an existing semantic segmentation architecture into the one suitable for real-time performance, while keeping the performance levels mostly intact. We achieved that by proposing simple modifications to the existing network and highlighting which building blocks were redundant for the final result. Our method can be applied along with any classification network for any dataset and can further benefit from using light-weight backbone networks, and other compression approaches. Quantitatively, we were able to closely match the performance of the original network while significantly surpassing its runtime and even acquiring 55 FPS on $512 \times 512$ inputs (from initial 20 FPS). Besides that, we demonstrate that having convolutions with large kernel sizes can be unnecessary in the decoder part of segmentation networks, and we will devote future work to further cover this topic.

**Acknowledgements** The authors would like to thank the anonymous reviewers for their helpful and constructive comments. This research was supported by the Australian Research Council through the Australian Centre for Robotic Vision (CE140100016), the ARC Laureate Fellowship FL130100102 to IR, and the HPC cluster Phoenix at the University of Adelaide.

# References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NIPS*, 2014.

[2] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *CoRR*, abs/1505.07293, 2015.

[3] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *ACM SIGKDD*, 2006.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[6] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016.

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611, 2018.

[9] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan L. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014.

[10] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[11] Dan C. Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, 2012.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[13] Gabriela Csurka and Florent Perronnin. A simple high performance approach to semantic segmentation. In *BMVC*, 2008.

[14] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[16] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. In *NIPS*, 2013.

[17] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

[18] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[19] Jun Fu, Jing Liu, Yuhang Wang, and Hanqing Lu. Stacked deconvolutional network for semantic segmentation. *CoRR*, abs/1708.04943, 2017.

[20] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, 2009.

[21] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[22] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[23] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014.

[24] Shixiang Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA*, 2017.

[25] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, 2013.

[26] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[27] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015.

[28] Bharath Hariharan, Pablo Arbelaez, Lubomir D. Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[29] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, 1992.

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

[31] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[32] 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. *CoRR*, abs/1704.04861, 2017.

[33] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[34] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[35] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[36] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[37] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[38] Stefan Kluckner, Thomas Mauthner, Peter M. Roth, and Horst Bischof. Semantic classification in aerial imagery by integrating appearance and height information. In *ACCV*, 2009.

[39] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

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

[41] Lubor Ladicky, Christopher Russell, Pushmeet Kohli, and Philip H. S. Torr. Associative hierarchical crfs for object class image segmentation. In *ICCV*, 2009.

[42] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *NIPS*, 1989.

[43] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *CVPR*, 2017.

[44] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[45] Guosheng Lin, Chunhua Shen, Ian D. Reid, and Anton van den Hengel. Efficient piecewise training of deep structured models for semantic segmentation. *CoRR*, abs/1504.01013, 2015.

[46] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.

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

[48] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen Change Loy, and Xiaoou Tang. Semantic image segmentation via deep parsing network. *CoRR*, abs/1509.02634, 2015.

[49] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[50] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[51] Volodymyr Mnih and Geoffrey E. Hinton. Learning to detect roads in high-resolution aerial images. In *ECCV*, 2010.

[52] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan L. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[53] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[54] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, 2016.

[55] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[56] Nils Plath, Marc Toussaint, and Shinichi Nakajima. Multi-class image segmentation using conditional random fields and global classification. In *ICML*, 2009.

[57] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[58] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[60] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[61] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[62] Jamie Shotton, John M. Winn, Carsten Rother, and Antonio Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

[63] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[64] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012.

[65] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[66] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *CoRR*, abs/1707.02968, 2017.

[67] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[68] Gregor Urban, Krzysztof J. Geras, Samira Ebrahimi Kahou, Özlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matthew Richardson. Do deep convolutional nets really need to be deep (or even convolutional)? *CoRR*, abs/1603.05691, 2016.

[69] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *ISCA*, 2016.

[70] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

[71] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *CVPR*, 2017.

[72] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[73] Jing Yang, Qingshan Liu, and Kaihua Zhang. Stacked hourglass network for robust facial landmark localisation. In *CVPR*, 2017.

[74] Fisher Yu, Vladlen Koltun, and Thomas A. Funkhouser. Dilated residual networks. In *CVPR*, 2017.

[75] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[76] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. *CoRR*, abs/1704.08545, 2017.

[77] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[78] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. *CoRR*, abs/1502.03240, 2015.

[79] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *CoRR*, abs/1702.03044, 2017.

[80] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *CoRR*, abs/1412.6856, 2014.

[81] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.

[82] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.