LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation

Abhishek Chaurasia
School of Electrical and Computer Engineering
Purdue University
West Lafayette, USA
Email: aabhish@purdue.edu

Eugenio Culurciello
Weldon School of Biomedical Engineering
Purdue University
West Lafayette, USA
Email: euge@purdue.edu

***Abstract*—Pixel-wise semantic segmentation for visual scene understanding not only needs to be accurate, but also efficient in order to find any use in real-time application. Existing algorithms even though are accurate but they do not focus on utilizing the parameters of neural network efficiently. As a result they are huge in terms of parameters and number of operations; hence slow too. In this paper, we propose a novel deep neural network architecture which allows it to learn without any significant increase in number of parameters. Our network uses only 11.5 million parameters and 21.2 GFLOPs for processing an image of resolution $3 \times 640 \times 360$. It gives state-of-the-art performance on CamVid and comparable results on Cityscapes dataset. We also compare our networks processing time on NVIDIA GPU and embedded system device with existing state-of-the-art architectures for different image resolutions.**

I. Introduction

Recent advancement in machines with ability to perform computationally intensive tasks have enabled researchers to tap deeper into neural networks. Convolutional neural networks' (CNNs) [1], [2] recent success has been demonstrated in image classification [3], [4], [5], [6], [7], [8], localization [9], [10], scene understanding [11], [12] etc. A lot of researchers have shifted their focus towards scene understanding because of the surge in tasks like augmented reality and self-driving vehicle, and one of the main step involved in it is pixel-level classification/semantic segmentation [13], [14].

Inspired by auto-encoders [3], [15], most of the existing techniques for semantic segmentation use encoder-decoder pair as core of their network architecture. Here the encoder encodes information into feature space, and the decoder maps this information into spatial categorization to perform segmentation. Even though semantic segmentation targets application that require real-time operation, ironically most of the current deep networks require excessively large processing time. Networks such as YOLO [16], Fast RCNN [17], SSD [18] focus on real-time object detection but there is very little to no work done in this direction in case of semantic segmentation [19].

In our work, we have made an attempt to get accurate instance level prediction without compromising processing time of the network. Generally, spatial information is lost in the encoder due to pooling or strided convolution is recovered by using the pooling indices or by full convolution. We hypothesize and later prove in our paper that instead of the above techniques; bypassing spatial information, directly from the encoder to the corresponding decoder improves accuracy along with significant decrease in processing time. In this way, information which would have been otherwise lost at each level of encoder is preserved, and no additional parameters and operations are wasted in relearning this lost information.

In Section III we give detailed explanation of our LinkNet architecture. The proposed network was tested on popular datasets: Cityscapes [20] and CamVid [21] and its processing times was recorded on NVIDIA Jetson TX1 Embedded Systems module as well as on Titan X GPU. These results are reported in Section IV, which is followed by conclusion.

II. Related work

Semantic segmentation involves labeling each and every pixel of an image and therefore, retaining spatial information becomes utmost important. A neural network architecture used for scene parsing can be subdivided into encoder and decoder networks, which are basically discriminative and generative networks respectively. State-of-the-art segmentation networks, generally use categorization models which are mostly winners of ImageNet Large Scale Visual Recognition Challenge (ILSCRC) as their discriminator. The generator either uses the stored pooling indices from discriminator, or learns the parameters using convolution to perform upsampling. Moreover, encoder and decoder can be either symmetric (same number of layers in encoder and decoder with same number of pooling and unpooling layers), or they can be asymmetric.

In [22] a pre-trained VGG was used as discriminator. Pooling indices after every max-pooling step was saved and then later used for upsampling in the decoder. Later on researchers came up with the idea of deep deconvolution network [23], [24], fully convolutional network (FCN) combined with skip architecture [25], which eliminated the need of saving pooling indices. Networks designed for classification and categorization mostly use fully connected layer as their classifier; in FCN they get replaced with convolutional layers. Standard pre-trained encoders such as: AlexNet [4], VGG [5], and GoogLeNet [7] have been used for segmentation. In order to get precise segmentation boundaries, researchers have also tried to cascade their deep convolutional neural network (DCNN) with post-

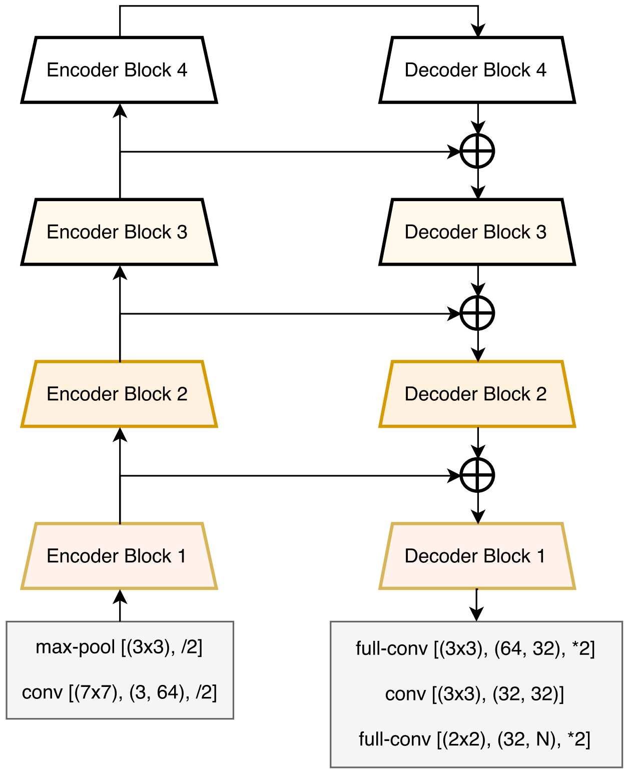


Fig. 1: LinkNet Architecture

processing steps, like the use of Conditional Random Field (CRF) [26], [11].

Instead of using networks which were designed to perform image classification, [27] proposed to use networks specifically designed for dense predictions. Most of these networks failed to perform segmentation in real-time on existing embedded hardware. Apart from this, recently recurrent neural networks (RNNs) were used to get contextual information [28] and to optimize CRF [29]; but the use of RNN in itself makes it computationally very expensive. Some work was also done in designing efficient network [30], [19], where DCNN was optimized to get a faster forward processing time but with a decrease in prediction accuracy.

III. Network Architecture

The architecture of LinkNet is presented in Figure [1]. Here, `conv` means convolution and `full-conv` means full convolution [25]. Furthermore, $/2$ denotes downsampling by a factor of 2 which is achieved by performing strided convolution,

TABLE I: Input and output feature maps

Block	Encoder		Decoder	
	m	n	m	n
1.	64	64	64	64
2.	64	128	128	64
3.	128	256	256	128
4.	256	512	512	256

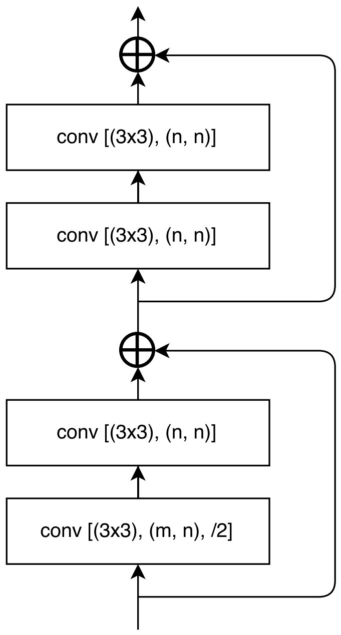


Fig. 2: Convolutional modules in *encoder-block (i)*

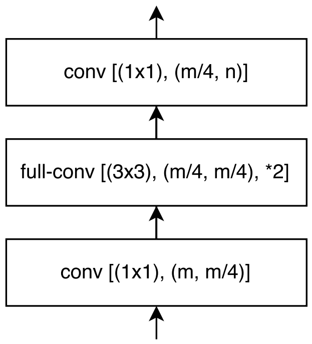


Fig. 3: Convolutional modules in *decoder-block (i)*

and $*2$ means upsampling by a factor of 2. We use batch normalization between each convolutional layer and which is followed by ReLU non-linearity [31], [32]. Left half of the network shown in Figure [1] is the encoder while the one the right is the decoder. The encoder starts with an initial block which performs convolution on input image with a kernel of size 7×7 and a stride of 2. This block also performs spatial max-pooling in an area of 3×3 with a stride of 2. The later portion of encoder consists of residual blocks [6] and are represented as *encoder-block(i)*. Layers within these *encoder-blocks* are shown in detail in Figure [2]. Similarly, layer details for *decoder-blocks* are provided in Figure [3]. Table [I] contains the information about the feature maps used in each of these blocks. Contemporary segmentation algorithms use networks such as VGG16 (138 million parameters), ResNet101 (45 million parameters) as their encoder which are huge in terms of parameters and GFLOPs. LinkNet uses ResNet18 as its encoder, which is fairly lighter network and still outperforms them as evident from Section

TABLE II: Performance comparison. Image size is W×H

Model	NVIDIA TX1						NVIDIA Titan X					
	480×320		640×360		1280×720		640×360		1280×720		1920×1080	
	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps
SegNet	757	1.3	1251	0.8	-	-	69	14.6	289	3.5	637	1.6
ENet	47	21.1	69	14.6	262	3.8	7	135.4	21	46.8	46	21.6
LinkNet	108	9.3	134	7.8	501	2.0	15	65.8	53	18.7	117	8.5

TABLE III: Comparison on the basis of operations

	GFLOPs	Parameters	Model size (fp16)
SegNet	286.0	29.5M	56.2 MB
ENet	3.8	0.4M	0.7 MB
Proposed Net	21.2	11.5M	22.0 MB

IV. We use the technique of full-convolution in our decoder as proposed earlier by [25]. Every conv$(k \times k)(im, om)$ and full-conv$(k \times k)(im, om)$ operations has at least three parameters. Here, $(k \times k)$ represent $(kernel - size)$ and (im, om) represent $(inputmap, outputmap)$ respectively.

Unlike existing neural network architectures which are being used for segmentation, our novelty lies in the way we link each encoder with decoder. By performing multiple downsampling operations in the encoder, some spatial information is lost. It is difficult to recover this lost information by using only the downsampled output of encoder. [22] linked encoder with decoder through pooling indices, which are not trainable parameters. Other methods directly use the output of their encoder and feed it into the decoder to perform segmentation. In this paper, input of each encoder layer is also bypassed to the output of its corresponding decoder. By doing this we aim at recovering lost spatial information that can be used by the decoder and its upsampling operations. In addition, since the decoder is sharing knowledge learnt by the encoder at every layer, the decoder can use fewer parameters. This results in an overall more efficient network when compared to the existing state-of-the-art segmentation networks, and thus real-time operation. Information about trainable parameters and number operations required for each forward pass is provided in detail in Section IV.

IV. Results

We compare LinkNet with existing architectures on two different metrics:

1) Performance in terms of speed:
   - Number of operations required to perform one forward pass of the network
   - Time taken to perform one forward pass
2) Performace interms of accuracy on Cityscapes [20] and CamVid [21] datasets.

A. Performance Analysis

We report inference speed of LinkNet on NVIDIA TX1 embedded system module as well as on widely used NVIDIA TitanX. Table II compares inference time for a single input frame with varying resolution. As evident from the numbers provided, LinkNet can process very high resolution image at 8.5 fps on GPU. More importantly, it can give real-time performance even on NVIDIA TX1. '-' indicates that network was not able to process image at that resolution on the embedded device.

We choose 640×360 as our default image resolution and report number of operations required to process image of this resolution in Table III. Number of operations determine the forward pass time of any network, therefore reduction in it is more vital than reduction in number of parameters. Our approach's efficiency is evident in the much low number of operations per frame and overall parameters.

B. Benchmarks

We use Torch7 [33] machine-learning tool for training with RMSProp as the optimization algorithm. The network was trained using four NVIDIA TitanX. Since the classes present in all the datsets are highly imbalanced; we use a custom class weighing scheme defined as $w_{\text{class}} = \frac{1}{\ln(1.02+p_{\text{class}})}$. This class weighing scheme has been taken from [19] and it gave us better results than mean average frequency. As suggested in Cityscapes [20], we use intersections over union (IoU) and instance-level intersection over union (iIoU) as our performance metric instead of using pixel-wise accuracy. In order to prove that the bypass connections do help, each table contains IoU and iIoU values with as well as without bypass. We also compare LinkNet's performance with other standard models such as SegNet [24], ENet [19], Dilation8/10 [34], and Deep-Lab CRF [35].

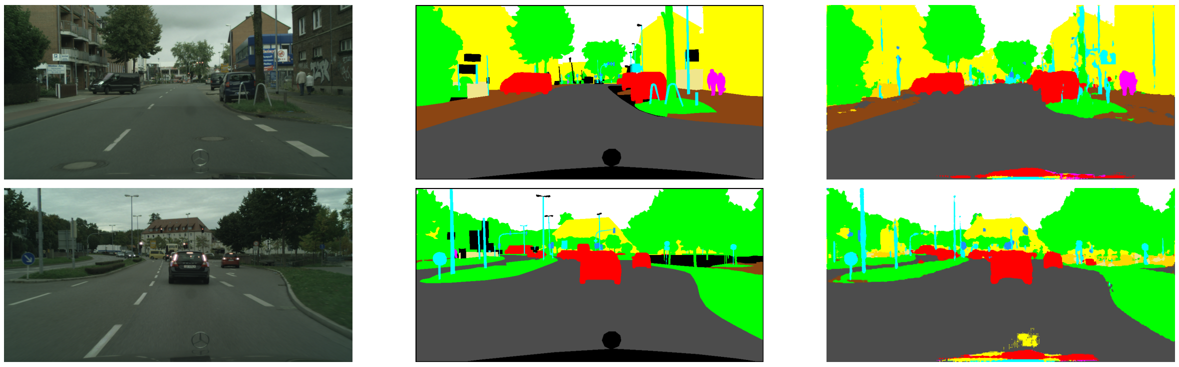
a) Cityscapes: This dataset consists of 5000 fine-annotated images, out of which 2975 are available for training, 500 for validation, and the remaining 1525 have been selected as test set [20]. We trained on our network on 19 classes that was provided in the official evaluation scripts [20]. As reported in Table IV, our network outperforms existing models.

TABLE IV: Cityscapes val set results (* on test set)

Model	Class IoU	Class iIoU
SegNet*	56.1	34.2
ENet*	58.3	34.4
Dilation10	68.7	-
Deep-Lab CRF (VGG16)	65.9	-
Deep-Lab CRF (ResNet101)	71.4	42.6
LinkNet without bypass	72.6	51.4
LinkNet	**76.4**	**58.6**

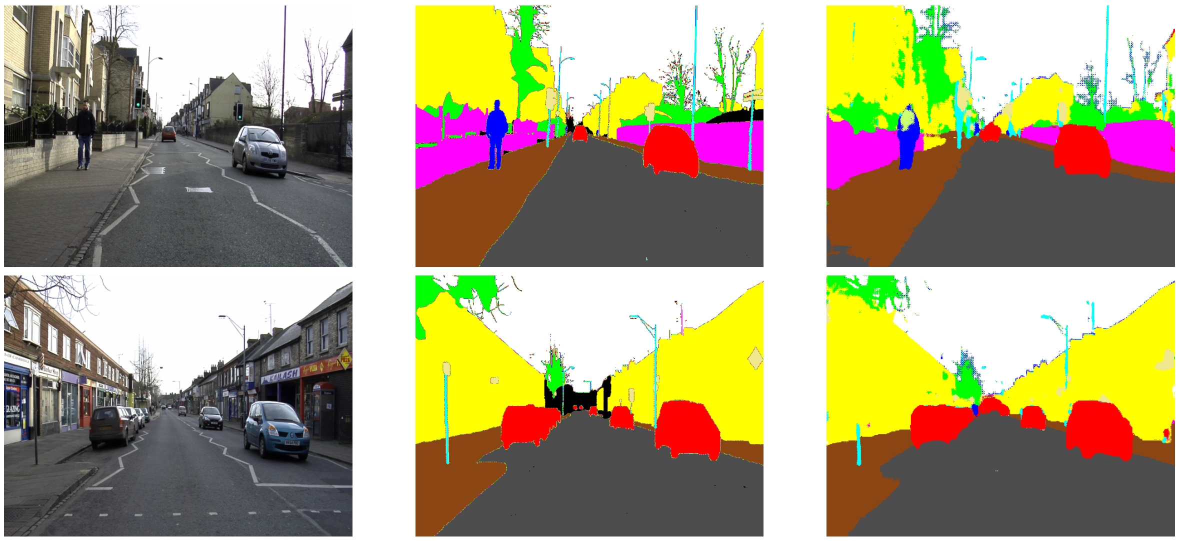
TABLE V: Results on CamVid test set of (1) SegNet, (2) ENet, (3) Dilation8, (4) LinkNet without bypass, and (5) LinkNet

Model	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Bicyclist	IoU	iIoU
1	88.8	87.3	92.4	82.1	20.5	97.2	57.1	49.3	27.5	84.4	30.7	65.2	55.6
3	74.7	77.8	95.1	82.4	51.0	95.1	67.2	51.7	35.4	86.7	34.1	68.3	51.3
2	82.6	76.2	89.9	84.0	46.9	92.2	56.3	35.8	23.4	75.3	55.5	65.3	-
3	84.6	87.4	88.8	72.6	37.1	95.3	61.2	56.0	33.1	88.3	24.4	66.3	52.7
4	88.8	85.3	92.8	77.6	41.7	96.8	57.0	57.8	37.8	88.4	27.2	**68.3**	**55.8**



(a) Input Image (b) Ground truth (c) Prediction

Fig. 4: LinkNet prediction on Cityscapes [20] test set.



(a) Input Image (b) Ground truth (c) Prediction

Fig. 5: LinkNet prediction on CamVid [21] test set.

These performance values were calculated on validation dataset. Input image of resolution 1024×512 was used for training the network. A batch size of 10 and initial learning rate of $5e{-}4$ was found to give optimum performance. Figure 4 shows the predicted segmented output on couple of cityscapes test images.

b) CamVid: It is another automotive dataset which contains 367 training, 101 validation, and 233 testing images [21]. There are eleven different classes such as building, tree, sky, car, road, etc. while the twelfth class contains unlabeled data, which we ignore during training. The original frame resolution for this dataset is 960×720 (W,H) but we downsampled the images by a factor of 1.25 before training. Due to hardware constraint, batch size of 8 was used to train the network. In Table V we compare the performance of the proposed algorithm with existing state-of-the-art algorithms on test set. LinkNet outperforms all of them in both IoU and iIoU metrics. Segmented output of LinkNet can be seen in Figure 5

V. Conclusion

We have proposed a novel neural network architecture designed from the ground up specifically for semantic segmentation. Our main aim is to make efficient use of scarce resources available on embedded platforms, compared to fully fledged deep learning workstations. Our work provides large gains in this task, while matching and at times exceeding existing baseline models, that have an order of magnitude larger computational and memory requirements. The application of proposed network on the NVIDIA TX1 hardware exemplifies real-time portable embedded solutions.

Even though the main goal was to run the network on mobile devices, we have found that it is also very efficient on high end GPUs like NVIDIA Titan X. This may prove useful in data-center applications, where there is a need of processing large numbers of high resolution images. Our network allows to perform large-scale computations in a much faster and more efficient manner, which might lead to significant savings.

Acknowledgment

This work was partly supported by the Office of Naval Research (ONR) grants N00014-12-1-0167, N00014-15-1-2791 and MURI N00014-10-1-0278. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TX1, Titan X, K40 GPUs used for this research.

References

[1] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, pp. 255–258, 1998.

[2] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, *Neural Networks: Tricks of the Trade*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, ch. Efficient BackProp, pp. 9–50.

[3] M. A. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 2007, pp. 1–8.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[8] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv:1602.07261*, 2016.

[9] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[10] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 648–656.

[11] P. Sturgess, K. Alahari, L. Ladicky, and P. H. Torr, "Combining appearance and structure from motion features for road scene understanding," in *BMVC 2012-23rd British Machine Vision Conference*, 2009.

[12] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.

[13] X. Ren, L. Bo, and D. Fox, "Rgb-(d) scene labeling: Features and algorithms," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 2759–2766.

[14] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, Aug 2013.

[15] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.

[19] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[20] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV (1)*, 2008, pp. 44–57.

[22] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *arXiv preprint arXiv:1505.07293*, 2015.

[23] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[24] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[26] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[27] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[28] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville, "Reseg: A recurrent neural network-based model for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 41–48.

[29] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[30] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[33] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, 2011.

[34] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[35] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915*, 2016.