Fast and Light-Weight Unsupervised Depth Estimation for Mobile GPU Hardware

Sangyun Oh[*1,2], Jongeun Lee[1,2], and Hye-Jin S. Kim[*3]

[1]School of Electrical and Computer Engineering, UNIST, Ulsan, Korea
[2]Neural Processing Research Center, Seoul National University, Seoul, Korea
[3]Creative Contents Reseach Division, SW/Contents Research Laboratory, ETRI, Daejeon, Korea
{syoh,jlee}@unist.ac.kr, marisan@etri.re.kr

Abstract

Recently, deep architectures for depth estimation have been proposed for many applications as depth is a very essential element of 3D geometry in computer vision. However, we still need to make the architectures efficient in terms of model size and computation complexity in order for them to be adopted in mobile environments. While model size has been reduced greatly current state-of-the-art compression techniques, the amount of computation has not been much. In this paper, we propose a lightweight architecture that also reduces computational complexity by adopting RR(Reduction and Reconstruction) block. The design principles are motivated by Tiny-Darknet which is one of image classification architecture. We basically refer to existing compression techniques, but limit maximum number of convolutional layer expansions that have been implemented for performance preservation. Through this, total amount of computation does not increase more than a certain amount. Our architecture shows significant architectural savings 30 $\sim$ 59$\times$ in the number of trainable parameters compared to existing architectures. Furthermore, we demonstrated our architecture on the mobile GPU hardware and affords from 3.88 $\sim$ 5.37$\times$ less energy consumption and 2.54 $\sim$ 3.11$\times$ faster runtime while allows only 1.66%p $\sim$ 2.08%p performance degradation compare to our baseline application.

1. Introduction

Depth estimation is a core problem for many computer vision applications, including AR/VR localization, 3D model reconstruction, and autonomous vehicles and UAVs and their applications extend to mobile environments in which there have been little consideration so far. Most depth estimation methods [3, 9, 15, 16] are focusing on stereo approaches, which achieve high performance. For mobile application, however, monocular approach [4, 6, 14] is more applicable and efficient way due to the number of cameras, energy consumption, memory usage, etc. although their accuracy is lower than that of stereo approaches. Among the monocular depth estimation methods, Godard [6] achieved the best performance in CVPR 2017 due to generating right side image from the left image with left-right consistency. We selected Godard method [6] as our baseline application. In terms of computational complexity and memory requirement, the CNN encoder-decoder architectures require huge resources because deeper structure or overlapping information from previous layers tends to improve their performance. This feature makes it difficult to efficiently apply to embedded hardwares and mobile as well.

Therefore, it is very important to reduce computational complexity and memory requirement of CNN encoder by appropriately adjusting weight parameters or the whole network architecture. The most intuitive way to reduce the number of weight parameters is using a very small-sized kernel window such as 1-by-1 or greatly limit the depth of output channels with appropriate sub-sampling. However, those can be disadvantageous in terms of performance because the absolute amount of trainable data can be strongly limited. Therefore, in prior works, after applying the above method, re-expansion is performed by various techniques [8, 10, 12, 13].

This paper makes the following contributions. First, we propose a lightweight CNN encoder suitable for embedded devices by considering both weight parameters and computational dilation. In particular, the encoder CNN is carefully designed so that Godard [6] can be executed on an embedded device, NVIDIA Jetson TX2 [1]. Second, we present our evaluation results which show that our model consumes 3.88 $\sim$ 5.37$\times$ less energy while running 2.54 $\sim$ 3.11$\times$ faster compared with original method [6] at the expense of only 1.66%p $\sim$ 2.08%p performance degradation.

This work was supported in part by The Cross-Ministry Giga KOREA Project grant from the Ministry of Science, ICT and Future Planning, Korea, and in part by Samsung Advanced Institute of Technology.

[*]These authors contributed equally to this work. *(Corresponding author: Hye-Jin S. Kim)*

Table 1: Our proposed network model (OFMs refers to output feature maps, IFM refers to input feature maps, K is the kernel size, and Rpt is a repetition in a RR block)

Layer	Type	OFMs	IFMs	K	Rpt
Conv1	forward	16	3	3	
Pool1	max-pool				
Conv2	forward	32	16	3	
Pool2	max-pool				
RRblk3	reduction	16	32	1	
	expansion	128	16	3	1
Pool3	max-pool				
RRblk4	reduction	32	128	1	
	expansion	256	32	3	1
Pool4	max-pool				
RRblk5	reduction	64	256	1	
	expansion	512	64	3	1
Pool5	max-pool				
Drop5	ratio: 0.7				
Conv6	forward	512	512	1	

2. Network Design

Depth estimation methods often consist of encoder-decoder model. Monocular depth estimation of [6] also uses a similar architecture combining well-known CNNs such as Alexnet, VGG, and ResNet. Those architectures are known for good performance in many applications but do not pay much attention to the memory, energy consumption and speed, which can be very important in mobile environments.

To overcome this problem, we designed a new model that does not go through multiple layer expansions, so computations will not increase significantly. Moreover, through design space exploration, we found that it is usually better to perform the above process repeatedly and up to 2 times of repeating can reach the target performance without significant increase in computation. We call the block that performs the above *RR block* (Reduction and Reconstruction block). Multiple RR blocks can be considered as needed. A sub-sampling layer is connected between blocks. We consider limiting kernel size to 3-by-3 to reduce the number of weight parameters and strongly restrict the depth of the output channels.

The architecture we propose is detailed in Table 1. There are three RR blocks and the repetition of re-expansion is set to 1, which is not the maximum number but it was found sufficient. In addition, to compensate for performance during training phase, a dropout layer was added with the ratio of 0.7.

3. Experimental Results

For evaluation we use a mobile GPU board, NVIDIA Jetson TX2 Development Kit [1]. It consists of quad-core ARM-A57 CPU and Jetson TX2 as a GPU with 8GB of main memory. We compiled Tensorflow 1.4.0 [2] from source for the ARM architecture and ported it to TX2 board with CUDA 8.0 and cuDNN 6.0 support. We evaluated our architecture using KITTI 2015 dataset [5].

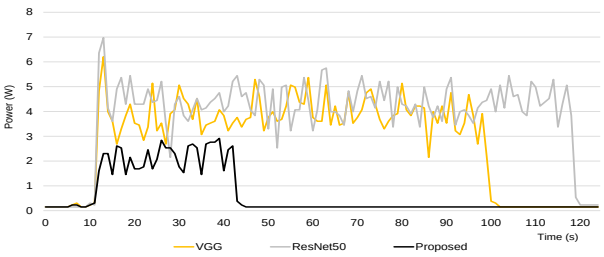

Figure 1: Energy consumption of NVIDIA Jetson TX2.

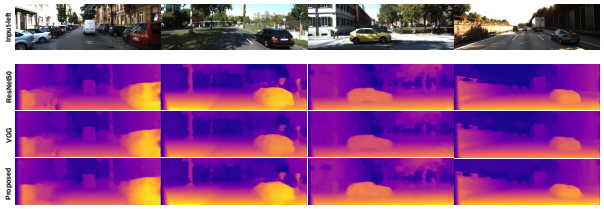

Figure 2: Depth estimation map results on KITTI 2015.

Table 2: Results on KITTI 2015

Application	monoDepth/ ResNet [6]	monoDepth/ VGG [6]	monoDepth/ Ours
Model size (MB)	670	362	12
FP ops* ($\times 10^9$)	19.40	2.24	0.49
D1-all (%)	8.78	9.20	10.86
Runtime (s)	0.33	0.27	0.10
Power (W)	4.05	3.59	2.34
Energy (J)	528.15	382.90	98.28

*FP operations are for the encoder part only.

As shown in Figure 1 and Table 2, Our model shows 3.88 $\sim$ 5.37$\times$ less energy consumption and 2.54 $\sim$ 3.11$\times$ faster while allowing only 1.66%p $\sim$ 2.08%p performance degradation compared with complex encoders [7, 11].

4. Conclusion

In this paper, we proposed a variant of models based on RR block model that is vastly improved in terms of speed and model size that it can be now run on resource-limited mobile GPUs such as TX2. Our design approach limits the total amount of computation and regulates reduction-expansion repetition, bearing in mind the overall computational complexity. When applied to monocular depth estimation, our evaluation shows that our proposed model can achieve multiple times improved results in terms of throughput and energy consumption while maintaining a similar level of performance.

References

[1] The nvidia jetson tx2. https://developer.nvidia.com/embedded/buy/jetson-tx2.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[3] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. *arXiv preprint arXiv:1803.08669*, 2018.

[4] R. F. David Eigen, Christian Puhrsch. Depth map prediction from a single image using a multi-scale deep network, 2014. Proceedings of NIPS.

[5] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of CVPR*, 2012.

[6] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778, 2016.

[8] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[9] G. Liang, Feng and Liu. Learning for disparity estimation through feature constancy, 2017. arXiv:1712.01039.

[10] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

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

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. CVPR, 2015.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.

[14] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *Proceedings of CVPR*, 2017.

[15] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:1–32, 2016.

[16] C. Zhou, H. Zhang, X. Shen, and J. Jia. Unsupervised learning of stereo matching. In *Proceedings of CVPR*, pages 1567–1575, 2017.