Double Refinement Network for Efficient Indoor Monocular Depth Estimation

*Nikita Durasov
Samsung AI Center Moscow,
Moscow Institute of
Physics and Technology
(State University)
n.durasov@samsung.com

*Mikhail Romanov
Samsung AI Center Moscow
m.romanov@samsung.com

*Valeriya Bubnova
Samsung AI Center Moscow,
Higher School of Economics
(State University)
v.bubnova@samsung.com

Anton Konushin
Samsung AI Center Moscow,
Moscow State University
a.konushin@samsung.com

Abstract

Monocular Depth Estimation is an important problem of Computer Vision that may be solved with Neural Networks and Deep Learning nowadays. Though recent works in this area have shown significant improvement in accuracy, state-of-the-art methods require large memory and time resources. The main purpose of this paper is to improve performance of the latest solutions with no decrease in accuracy. To achieve this, we propose a Double Refinement Network architecture. We evaluate the results using the standard benchmark RGB-D dataset NYU Depth v2. The results are equal to the current state-of-the-art, while frames per second rate of our approach is significantly higher (up to 15 times speedup per image with batch size 1), RAM per image is significantly lower.

1. Introduction

Monocular Depth Estimation is an important problem of Computer Vision that may be solved with Neural Networks and Deep Learning nowadays. Besides the interest in the Depth Estimation problem itself for mobile photography and other applications, the network that can produce good depth estimations may be used as a backbone for such problems as Simultaneous Localization and Mapping (SLAM), Visual Odometry (VO), Augmented Reality applications and many others. It also may be used as a base model for the better Stereo Depth Estimation.

Unlike Stereo Depth Estimation, this approach implies depth reconstruction from a single image and does not require usage of stereo cameras. That is the reason why monocular depth estimation is a perspective method for industry uses. One of the examples may be Computer Vision in Robotics, where stereo capturing is associated with such difficulties as non-portable setting and extra expenditures.

However, the main obstacle while solving 2D Depth Reconstruction problem is ambiguity of depth estimation without prior knowledge about sizes and focal length. That seems to be one of the reasons why deep learning methods are popular for solving Monocular Depth Estimation problem.

Though recent works in this area have shown significant improvement in accuracy, we believe that there is still some space for improvement. One of the possible areas for research is memory and time consumption. Typically, such problems are being solved with Deep Convolutional Neural Networks, which succeed in depth detection given only the image. However, along with convolutional layers, big amount of bilinearly interpolated deep feature maps play important role in the architecture. Such layers require large memory and time resources. That is one of the most important drawbacks of the latest state-of-the-art solutions. The main purpose of this paper is to improve performance of the latest solutions with no decrease in accuracy. We evaluate the results on standard benchmark RGB-D dataset NYU Depth v2.

The problem is stated as a regression problem of pointwise depth reconstruction. Ground truth for $H \times W \times 3$ image is represented as $H \times W \times 1$ depth map. In other words, we have to make depth estimation for every pixel.

The paper plan is the following: in the next section we will overview recent works addressing the problem of Monocular Depth Estimation, section 3 contains the de-

*These authors contributed equally to this work

scription of the proposed solution and the experiments can be found in Section 4. Finally, we summarize the paper in Section 5.

2. Related Work Overview

2.1. Before Deep Learning

The Depth Estimation is not a new research field. Scientists were interested in this area before Deep Learning became a leading technology in Computer Vision area. And even though Deep Learning took over the traditional Computer Vision around 2012-2013, for a long time the dominating methods in Depth Estimation were based on other techniques.

A widely known algorithm was proposed by Saxena et al. [21]. They use Markov Random Field for oversegmentation of the image. "Superpixels" obtained by the algorithm are used for building a 3D structure of the image by inferring 3D position and orientation of the 3D surface that it came from. Geometry-based algorithms for indoor Depth Estimaiton were introduced in [9] and [5]: they base their models on several assumptions about object orientation in a room with a known shape and some prior knowledge of usual objects placement in such rooms. Gupta et al. use SVM to compute a feature vector for each surface. [10] presented a system that models geometric classes that depend on the orientation of a physical scene and build the geometric structure progressively. Geometry-based approaches work with certain constraints and are relevant for only narrow type of images, such as images of a regular room in a house.

2.2. Deep Learning Era

Most recent works on Depth Reconstruction are based on Deep Convolutional Neural Networks [3, 14, 15, 16, 18]. It is a common practice to use pretrained layers (ex. VGG [22], SENet [12] or ResNet [7]) for feature extraction. Another noticeable trend is modifying encoder-decoder architectures. In such architectures convolutions and max-pooling layers are widely used for implementing encoder and upsampling layers often play role of a decoder. Here we overview the latest works with the highest benchmark results.

In [6] Gurram et al. proposed a solution which includes training a model with use of two heterogeneous datasets, which are datasets with depth and semantic map labels. Common NN layers are trained alternately on data from either of the datasets. In particular, the network consists of two flows that can be used for getting depth or semantic map of an image respectively.

In the work by Fu et al. [4] DCNN architecture performs Monocular Depth Estimation formulated as ordinal regression problem, i.e. it works with discretized depth. The architecture includes dense feature extractor (convolutions and dense layers), scene understanding modular (concatenation of dilated convolutions, full-image encoder and convolutions layers) and ordinal regression block.

The ambiguity caused by focal length mentioned above was studied by He et al. [8]. They suggested that using datasets with various focal lengths may be helpful in handling this issue. The proposed method implies generating multiple images with different focal lengths from one fixed-focal-length image. Thus, the model is trained on varying-focal-length dataset. Embedded focal lengths is given to a network as well as the images.

We should also mention another network that does Depth Estimation in real-time presented by Spek et al. [23].

2.3. Main Competitor

We base on the work of Hu et al. [11]. This method is based on utilizing residual network activation maps from all the network layers interpolated to the size of the input image concatenate next to the extra final activation maps of the custom decoder. This network shows good performance on the majority of the common metrics.

3. Method

We improve the Main Competitor's solution by introducing the following:

- New Double Refinement Network Architecture
- Auxiliary Losses

3.1. Double Refinement Network Architecture

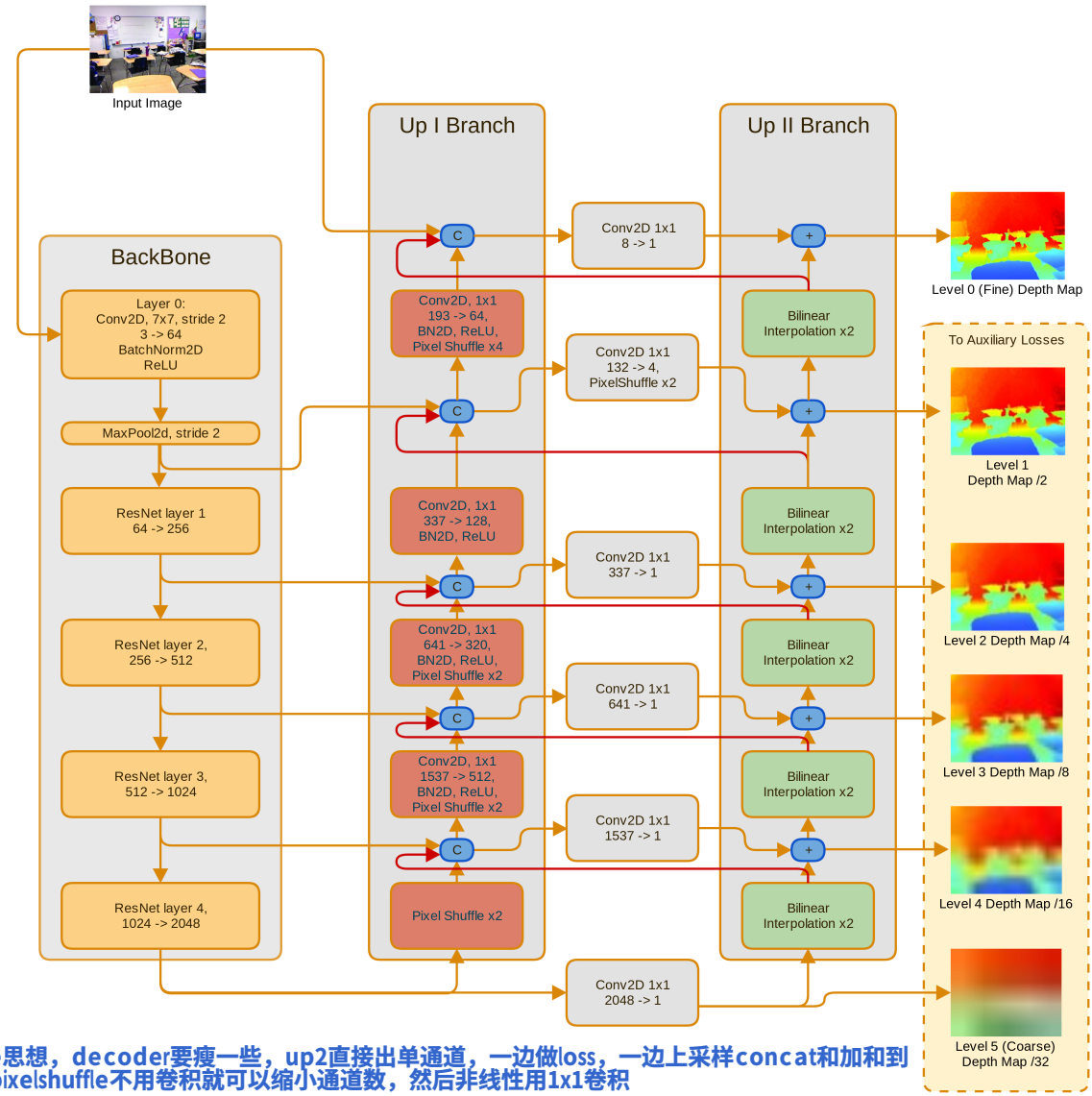
The architecture of the proposed network can be seen at Figure 1.

3.2. Drawbacks of State-of-the-Art Architectures

Current State-of-the-Art architectures are very time and memory consuming. Some of the architectures (such as Main Competitor's architecture) use Bilinear Interpolation of the deep activation maps. This procedure is very memory hungry. Talking about standard ResNet (>50) architecture applied to an image of size 224×224, deep layers give $64 \times 112 \times 112$, $256 \times 56 \times 56$, $512 \times 28 \times 28$, $1024 \times 14 \times 14$, $2048 \times 7 \times 7$.

Now, in case we follow the Main Competitor's strategy, we have to upsample all of the activation maps to half of the size of the original image and concatenate results. The resulting tensor will contain $3904 \times 224 \times 224$ values! More than that, it is necessary to make some operations with this tensor to get the depths. This leads to low fps rate.

Speaking about other segmentation networks that are also popular for depth estimation problem, bilinear interpolation is also used in those architectures. Besides, PSP-Net



类似于coarse2fine思想，decoder要瘦一些，up2直接出单通道，一边做loss，一边上采样concat和加和到下一层的输出上，pixelshuffle不用卷积就可以缩小通道数，然后非线性用1x1卷积

Figure 1. Double Refinement Network architecture. It contains one downsampling branch based on the backbone (yellow branch) and two upsampling branches(red and green branches). Red branch (`up I`) merges the high-level features together with low-level features. The upsampling in it is done by Pixel Shuffle. The green branch (`up II`) predicts only the Depth Map. In the right branch the upsampling is performed by Bilinear Interpolation. The `up II` branch outputs a depth approximation on each level, thus we can compute loss function on every level and force the high layers to learn as efficiently as low levels. The arrows that are marked with red color are referred to as diagonal connections in the text.

[24] and DeepLab v3 [1], [2] use dilated convolutions that are also memory and time consuming. The only architecture that gives good fps rate is the RefineNet architecture [17] that may be reinterpreted as U-net based on ResNet.

In our architecture we get rid of dilated convolutions and bilinear interpolation of big amounts of activation maps. Instead, we do Pixel Shuffle upsampling of the high-level features. But in case we do only this then we loose in accuracy. To avoid this, we follow the logic that the high-level outputs are profitable for coarse structure, while the low-level outputs should only refine coarser guess. To compute the correction, we need less information. Thus, we do not need

to make bilinear upsampling of deep tensors. Thus, we iteratively improve our coarse guess for the depth from layer to layer (we increase the depth map side by factor of 2 for each layer) making it finer and finer without doing time and memory complex operations.

3.2.1 Architecture Description

While reading this part we strongly recommend to keep an eye on the Figure 1.

We base our network on the Residual Network architecture introduced by He et al. [7]. This architecture is known for its high results on the ImageNet dataset. More specifically, we take pretrained ResNet-50, ResNet-152, DenseNet-161 [13] architectures from the PyTorch torchvision library [19] and SENet-154 [11].

The schema on the figure above is designed for pretrained ResNet $\geq$ 50 backbone, but the logic for the other networks is the same. The network consists of one downsampling branch (BackBone) and two up branches. We will use the following notation: $\texttt{down}_i$ is a result of the `down` branch on the level i. For the upsampling branches we will count from the top, i-th block of the `upI` branch we will denote as $\texttt{upI}_i$.

We take the pretrained network as is, cutting off only the last Fully Connected layer and AveragePooling layer. Thus, we get the backbone for our network that we build on. The network is split into several residual blocks (`layer 1,2,3` and `4`) plus several preprocessing elements:

- `Conv2D(7 × 7, 3 → 64, stride 2)`
- `BatchNorm2D(64)`
- `ReLU`
- `MaxPool2D(3 × 3, stride 2)`

This preprocessing block we denote as `layer 0`. Note that the preprocessing block downsamples the image's side 4 times.

Then the `down` branch of the network can be represented as:

- `layer 0`

  ...

- `layer 4`

We will denote the input image as `input`, the output of the `layer 0` as $\texttt{down}_1$, the output of `layer i` as $\texttt{down}_{i+1}$. We do the forward propagation through the BackBone branch as it is done in the backbone networks.

The next step is encouraged by the U-Network architecture introduced by Ronneberger et al. [20] and RefineNet.

To start describing the upsampling process, we denote $\texttt{down}_5$ as $\texttt{upI}_5$ for convenience.

To get the most coarse estimation of the depths $\texttt{upII}_5$ we apply `Conv2D` 1×1 to the $\texttt{upI}_5$.

This coarse depth approximation takes into account only the high-level outputs (note that in this output high-level features are entangled with low-level features due to the residual connections).

Now to get the $\texttt{upI}_i$, we take $\texttt{upI}_{i+1}$ tensor, performing the following operations to it:

- `Conv2d 1×1` (except $\texttt{upI}_5$ block)
- `BatchNorm2D` (except $\texttt{upI}_5$ block)
- `ReLU` (except $\texttt{upI}_5$ block)
- `PixelShuffle ×2` (except $\texttt{upI}_1$ block; for $\texttt{upI}_0$ block – `PixelShuffle ×4`).

The output is concatenated with depth approximation $\texttt{upII}_{i+1}$ upsampled by factor of 2 using bilinear interpolation.

The depth approximation $\texttt{upII}_i$ is the coarser depth $\texttt{upII}_{i+1}$ upsampled by factor of 2 using Bilinear Interpolation plus the correction. The correction is computed from $\texttt{upII}_i$ concatenated with downsampled image and upsampled coarser depth using `Conv2D 1×1` with one output activation.

It is worth noting that the second upsampling branch of the network `upII` is inspired by Fourier Transform. We first make a coarse prediction and then, ascending from the deepest levels to the shallow ones, we make the corrections to those predictions. Each level increases the side of an image by a factor of 2. To get the finest depth correction, we take the input image, concatenate it with 4 activation maps and depth $\texttt{upII}_1$ and apply a convolution.

3.2.2 Relation to the Main Competitor's Network

The Double Refinement Network architecture can be converted easily to the network from the work [11] by doing the following:

- Substitute the upsampling blocks in `upI` with Bilinear Interpolation
- Remove the `upII` branch of the network, leave only the finest output.
- Add extra decoder subnetwork.

From this we can see that doing some simple manipulations, we can come back to the network that gives the current State-of-the-Art result.

	RMSE	Log10	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
Laina et al. [16]	0.573	0.055	0.811	0.953	0.988
Fu et al. [4]	**0.509**	0.051	0.828	0.965	0.992
Spek et al. [23]	0.687	0.161	0.704	0.917	0.977
RSIDE ResNet-50 (paper)	0.555	0.054	0.843	0.968	0.991
RSIDE DenseNet-161 (paper)	0.544	0.053	0.855	0.972	0.993
RSIDE SENet-154 (paper)	0.530	**0.050**	0.866	**0.975**	0.993
RSIDE SENet-154 (reproduced)	0.574	0.052	0.845	0.967	0.991
(ours) DRNet ResNet-50	0.587	0.061	0.810	0.961	0.990
(ours) DRNet ResNet-152	0.528	0.050	0.866	0.973	0.993
(ours) DRNet DenseNet-161	0.534	0.052	0.865	0.974	**0.994**
(ours) DRNet SeNet-154	0.527	**0.050**	**0.868**	**0.975**	0.993

Table 1. Comparison of our network with different backbones against Revisiting Single Image Depth Estimation's (RSIDE) [11] and others' results. It is observable that our network gives around the same result (it is sometimes a bit higher though). We should note that with ResNet-50 backbone our network does not reach the same results as in the Main Competitor's work. Possibly, this can be fixed by doing more accurate fine-tuning.

	fps (b 1)	fps (max b)	net RAM	RAM / img
RSIDE ResNet-50 (reproduced)	2	2 (b 1)	793 Mb	21 Gb
RSIDE DenseNet-161 (reproduced)	1.7	1.7 (b 1)	853 Mb	22 Gb
RSIDE SENet-154 (reproduced)	1.5	1.5 (b 1)	1.7 Gb	21.9 Gb
(ours) DRNet ResNet-50	36	69 (b 36)	757 Mb	2 Gb
(ours) DRNet ResNet-152	25	55 (b 19)	1067 Mb	2.9 Gb
(ours) DRNet DenseNet-161	17	33 (b 14)	839 Mb	3.2 Gb
(ours) DRNet SeNet-154	6	13 (b 9)	1421 Mb	4 Gb

Table 2. Comparison of our network with different backbones against the Revisiting Single Image Depth Estimation's (RSIDE) [11] fps and memory. It is observable that our network uses less RAM per image and fps rate is higher. FPS rates are measured using Tesla P40 GPU. In this table "b" stands for the batch size.

But doing bilinear interpolation of the residual blocks' activation maps to the size of the original image, as it is done in the Main Competitor's work, is suboptimal as this procedure is computationally complex, the resulting tensor is large and does not change much from pixel to pixel (especially when we interpolate the `level 4` block's features).

3.2.3 Auxiliary Loss Functions

We have added the auxiliary losses on each of the outputs. We take the i-th level depth and penalize it as we do it for the finest depth approximation (the target is downsampled 2^i times).

Thus, the overall loss function is computed as:

$$\tilde{\mathcal{L}} = \sum_{level=0}^{5} \mathcal{L}_{level}. \tag{1}$$

We downsample the target depth using bilinear interpolation to compute the level-specific loss. We have also tried upsampling the level-specific result using bilinear interpolation – and that worked a little bit worse than downsampling the target.

3.3. Loss Function

Depth estimation requires not only pixel-wise accuracy, but also spatially coherent result. That is the reason why many models use losses that includes depth gradient and normals. The loss we use consists of four parts:

$$\mathcal{L} = \mathcal{L}_{depth} + \mathcal{L}_{grad} + \mathcal{L}_{normal}. \tag{2}$$

These loss functions were already presented in the Main Competitor's work, we take them as is.

4. Results

We have done our experiments on the NYUv2 dataset. We based our code on the Main Competitor's code [1] in order to show that our network gives the reported results in *exactly* the same settings and to avoid result deviations caused by unrelated code changes.

We use the following common metrics to compare our model to the State-of-the-Art model:

[1] `https://github.com/JunjH/Revisiting_Single_Depth_Estimation`

	RMSE	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
DRNet ResNet-152	0.528	0.866	0.972	0.993
DRNet ResNet-152 no diagonal connections	0.533	0.866	0.973	0.993
DRNet ResNet-152 no auxiliary losses	0.544	0.860	0.971	0.992
DRNet ResNet-152 no second branch	1.900	0.351	0.596	0.732

Table 3. Reasoning of new architecture's elements. As we can see, diagonal connections do not improve the result, auxiliary losses improves the result a bit. The biggest improvement gives the presence of the second upsampling branch.

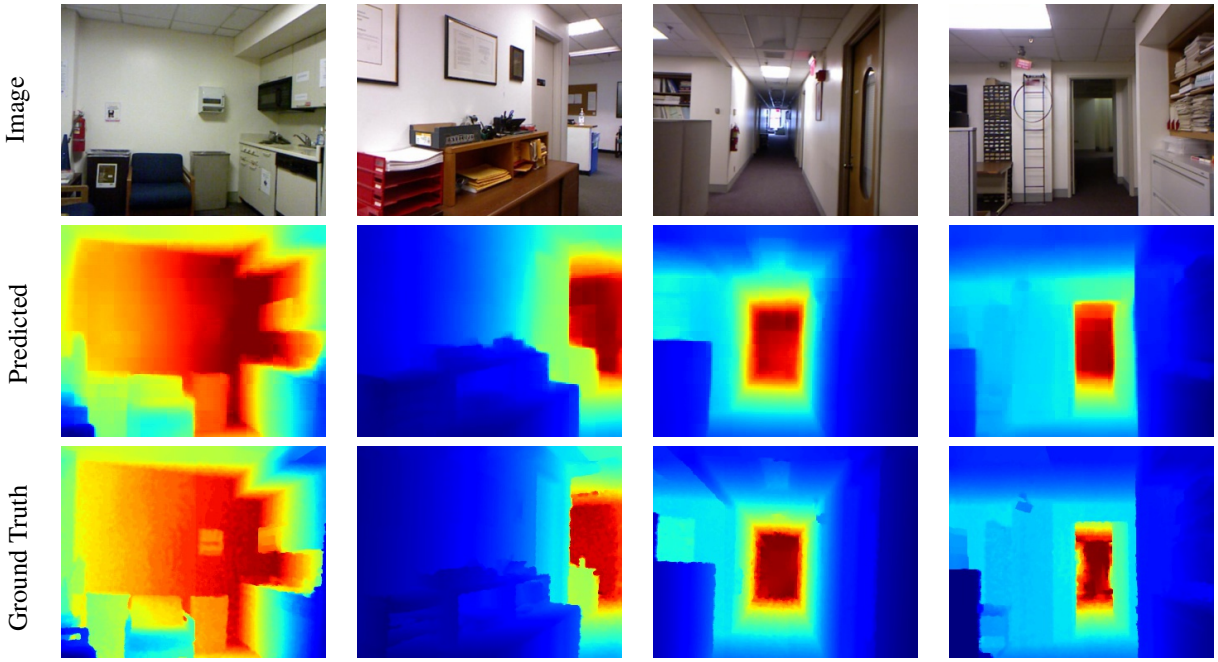


Table 4. Network predictions compared to the ground truth.

- Root mean squared error:

$$\sqrt{\frac{1}{|N|}\sum_{i \in N} |d_i - d_i^*|^2} \tag{3}$$

- Log 10 error:

$$\frac{1}{|N|}\sum_{i \in N} |\log_{10}(d_i) - log_{10}(d_i^*)| \tag{4}$$

- Threshold (δ):

$$\% \text{ of } d_i \text{ s.t. } \max\left(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}\right) < t, \tag{5}$$

where $t \in 1.25, 1.25^2, 1.25^3$

We test our network against the Main Competitor's network and some other related works. We thank our Main Competitor for sharing the original reproducible and clear code.

We train the network in the following way: we take pretrained backbone network. The training is performed using Adam algorithm (amsgrad modification) with learning rate 10^{-4}, weight decay 10^{-4}. The betas are selected to be default.

From the numerical results (see Table 1, 2) it is possible to see that the new architecture gives close metrics to the Main Competitor's architecture values with higher fps rate. Also, the new architecture consumes significantly less memory per image due to the fact that we do not make *that* many bilinearly upsampled feature maps. The visual results can be found in the Table 4.

We should note that during our experiments we used to get a bit higher values than the Main Competitor's architecture. We doubt that those deviations are statistically important. More than that, we should note that in the dataset there are many errors in the density maps. For example, in many cases the straight lines that are straight on the images are not straight on the depth maps. This causes the estimated depth maps to be blurry and very inaccurate. Possibly is a

good idea to train the depth estimation on a simulated data.

4.1. Element Reasoning

In our approach we have introduced several new elements to the network. To be sure that every element that we introduce improves the result, we do element reasoning experiments on the Table 3.

As we can see, the largest improvement gives the second upsampling branch, thus this element is the most important in the presented architecture. The auxiliary losses improve result slightly, but throughout our experiments there was no cases when it without those losses metrics were higher. Thus, auxiliary losses are also an important element. The diagonal connections' efficiency is under question as it sometimes improve metrics a little bit, but sometimes it doesn't. In cases when diagonal connections improve the result – the improvement is very small, when it doesn't – the losses in metrics are very small. Thus we have to admit that it is possible that those connections are useless.

5. Summary

We have introduced a new architecture for Monocular Depth Estimation. The network works significantly (15 times) faster on single image, gives Real-Time fps rate, uses 10 times less memory per image. Plus it gives the results comparable to the current State-of-the-Art. We have shown which parts of our network improve the result and how big the improvement is.

References

[1] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[2] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[3] 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*, pages 2650–2658, 2015.

[4] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[5] A. Gupta, M. Hebert, T. Kanade, and D. M. Blei. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *Advances in neural information processing systems*, pages 1288–1296, 2010.

[6] A. Gurram, O. Urfalioglu, I. Halfaoui, F. Bouzaraa, and A. M. Lopez. Monocular depth estimation by learning from heterogeneous datasets. *arXiv preprint arXiv:1803.08018*, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] L. He, G. Wang, and Z. Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 2018.

[9] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *European Conference on Computer Vision*, pages 224–237. Springer, 2010.

[10] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 654–661. IEEE, 2005.

[11] J. Hu, M. Ozay, Y. Zhang, and T. Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. 2018.

[12] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. 2018.

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] Y. Kuznietsov, J. Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6647–6655, 2017.

[16] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[17] G. Lin, A. Milan, C. Shen, and I. D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, volume 1, page 5, 2017.

[18] F. Liu, C. Shen, G. Lin, and I. D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):2024–2039, 2016.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[20] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[21] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[23] A. Spek, T. Dharmasiri, and T. Drummond. Cream: Condensed real-time models for depth prediction using convolutional neural networks. *arXiv preprint arXiv:1807.08931*, 2018.

[24] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.