Light-Weight Learning-Based Depth Estimation From A Single Image

Zekarias Negese[1], Hartmut Bauermeister[2], Michael Moeller[2], Emanuele Rodolà[1], Zorah Lähner[2]

[1] Sapienza Università di Roma, Dipartimento di Informatica, Via Salaria 113, 00198 Roma, Italy
znegese@gmail.com, rodola@di.uniroma1.it

[2] University of Siegen, Department für Elektrotechnik und Informatik, Hölderlinstr. 3, 57076 Siegen, Germany
{hartmut.bauermeister, michael.moeller, zorah.laehner}@uni-siegen.de

Abstract. *Estimating the depth of a scene from a single image is impossible due to scale ambiguity, but recent deep learning approaches have demonstrated to produce faithful estimates anyways by learning the typical scale of objects from implicit clues in the scene. In this paper, we present a new encoder-decoder architecture for single image depth reconstruction that yields state-of-the-art results with considerably less parameters than competitors. Our architecture incorporates a modified inception module tailored to our application which allows us to use a relatively lightweight architecture with fewer learnable parameters than state-of-the-art single image depth reconstruction networks. We train and evaluate on the known NYU RGB-D benchmark dataset, and show generalization of our method by evaluating the pretrained NYU model on the iBims dataset. Our results are in the same order of magnitude as state-of-the-art competitors, even though we trained on a different resolution.*

Keywords. ATHENA Research Book, Monocular Depth, Deep Learning, CNNs, Efficient Deep Learning, Inception Module

1 Introduction

Classical approaches to capture the geometry of a scene require at least two viewpoints (e.g. a stereo [1] or light field camera [2]), a sequence of images, such as focal

stacks to extract depth clues from the depth of field in depth-from-focus (e.g. [3]) and depth-from-defocus (e.g. [4]) approaches, or special recording techniques with active lighting such as structured light or time-of-flight cameras. Applications of such techniques range from image refocusing, over the initialization of 3D reconstruction algorithms, e.g. in autonomous driving, to augmented reality. However, estimating the depth of a scene from only a single RGB image is an inherently ill-posed problem, which – from a purely physical perspective – cannot distinguish between the size of an object and its distance to the camera. To solve this problem, prior knowledge of typical object sizes as well as blur focus at different depth levels can give meaningful clues about the scene composition and depth. For a certain type of natural images, numerous recent works have exploited deep learning techniques to still make faithful predictions, demonstrating that implicit clues of the size of everyday objects are often sufficient to still provide accurate estimates, see Section 2.

At the same time, more powerful hardware allowed training incredible complex deep learning models with ever improving results. Complex models with a huge amount of parameters are able to learn very complicated functions but naturally also require a larger amount of memory and computing power, even at inference time. This is often not feasible on smaller or cheaper hardware that is used, for example, in many mobile applications and consumes, sometimes unnecessary, large amounts of electricity. The purpose of this paper is to introduce a network architecture for monocular depth estimation which balances network size and training time without sacrificing too much accuracy or overfitting to specific training data.

To this end, we propose an encoder-decoder network for monocular depth estimation using inception style layers [5]. Our architecture performs on-par with state-of-the-art algorithms while being comparably light-weight, i.e. having less trainable parameters, and is generalizable to different datasets and resolutions.

Contributions. We propose a novel encoder-decoder architecture for depth from a single RGB image using inception blocks. Our architecture optimizes the trade-off between having few parameters for efficient training and inference but still resulting in good accuracy. The multi-scale property of this block as well as its separation of channel and spatial dimensions lead to convincing performance on the NYU dataset [6]. In addition to yielding accurate results, our architecture has considerably fewer trainable parameters than direct competitors which makes it more efficient in both training as well as inference. We show that our results are generalizable to different resolutions and datasets by evaluating the pretrained NYU model on the iBims dataset [7].

2 Related Work

In this section, we review existing work for monocular depth reconstruction that is directly related to our method (Section 2.1), as well as work about increasing the efficiency of neural networks (Section 2.2). We refer the interested reader to [8] for an in-depth overview of deep learning based monocular depth estimation.

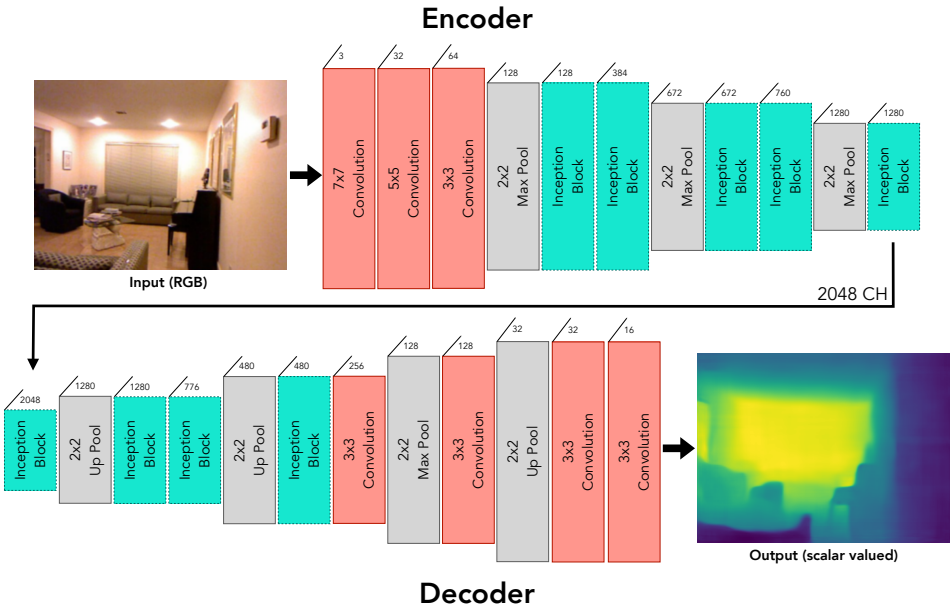

Figure 1. The proposed model architecture. We employed an inception style encoder-decoder network. The layers with red colors indicate basic spatial convolution, while the layers with cyan color indicate inception layers (also see Figure 2). The layers with gray color indicate max-pooling layers in the encoder part and up-sampling layers in the decoder layer. The number over each block indicate the input feature dimension. Exact details of the architecture are described in Table 5.

2.1 Monocular Depth

Predicting depth from a single image is an ill-posed problem in which the influence of scale and distance to the camera cannot be completely separated. As a result, the majority of literature about depth estimation considers settings with at least stereo vision [9], or coming directly from specialized hardware [10]. However, reasonable estimations can be made with semantic prior knowledge about scenes. With the rise of deep learning, using huge amount of training data became feasible and, as a result, made solving this problem possible. One of the first major works in this direction was [11]. They proposed a single image method for depth estimation by training two consecutive CNNs. The first CNN learns coarse global information based on the whole image and then the second deep network refines it by learning local cues. [12] also used a two stage strategy where they first categorized the RGB image into different RGB ranges using a scene understanding module, and then in the second stage, train a network on the images with a specific depth range. Similar works on depth estimation from a single image [13]–[18] have been proposed based on CNNs with different architectures and loss functions. [19] proposed to train an encoder-decoder CNN where the encoder part employs DenseNet-169.

Other directions include more geometric information directly into the network architecture. [20] described a depth estimation technique using local planar guidance layers placed at multiple stages in the decoding phase of an encoder-decoder network. [21] proposed a monocular depth estimation method with stable geometric constraints from a global perspective to take long-range constraints into account, termed as virtual normals. [22] proposed a novel technique called attention based context aggregation network (ACAN). They adopt a deep residual network [23] where dilated convolutions are used to maintain the spatial scale. [24] proposed SharpNet, a method that predicts an accurate depth map from given a single input color image, with particular attention to the reconstruction of occluding points, by constraining the depth estimation and occluding contours during model training.

Current state-of-the-art is able to produce highly accurate depth maps but at the price of very complex networks with long training times (see Table 4). Our work is also a CNN encoder-decoder, but instead of explicitly including expensive geometric priors into the architecture we focus on the higher priority of cross-channel relationships, as opposed to spatial information in normal convolutions, by using inception blocks [5]. We show that this prior is powerful enough for monocular depth to achieve results on par with state-of-the-art using considerably less parameters than previous methods. Chen et al. [25] already build a network with inception blocks for this problem with great success in reducing the parameters but at the cost of accuracy.

2.2 Efficient Networks

With more powerful GPUs available in the last years, it became possible to train larger and larger networks with some methods taking several days to train architectures with billions of parameters [26]. While this often leads to superior results, the energy consumption rises and some hardware can only deploy smaller networks [27]. This is especially important for mobile and IoT applications where both are limited. One of the first major papers to tackle this was [28]. The method uses depth-wise separable convolutions to manage the size of the network, and lets the user adjust the performance-complexity ratio through hyperparameters. Many works followed up on this, putting the focus mainly in classification on mobile applications [29]–[31].

[16] worked on reducing the complexity of monocular depth estimation architectures. While producing a very small network with fast inference time, their accuracy is not on-par with state-of-the-art results. [32] proposed a monocular depth estimation for IoT devices which are normally even more restricted than mobile hardware. Therefore, this work focuses on energy consumption instead of accuracy, and only works on restricted image resolutions. Our method goes in the direction of [33] but for monocular depth instead of general convolutional nets: reducing the complexity of the network without a specific hardware in mind while achieving state-of-the-art results.

3 Method

In this work, we employ an encoder-decoder CNN architecture with inception blocks. The architecture is explained in detail in the following sub-sections. The entire network architecture is shown in Figure 1 and Table 5 in the supplementary. Instead of an encoder-decoder network with basic convolutional layers, an inception-like encoder-decoder network is used to get the advantage of multi-level feature extraction in both spatial and cross-channel dimensions.

The inception blocks are a key feature that allow our architecture to consistently perform on-par with state-of-the-art methods while using considerably fewer parameters than any competitor. Due to a collection of 1×1 convolutions (see Section 3.1), inception blocks put more weight on cross-channel information than on spatial relations. We claim that this induces a more meaningful and stronger network prior for the task of depth prediction on natural images than existing models, and this is the reason for our good performance with a smaller network architecture.

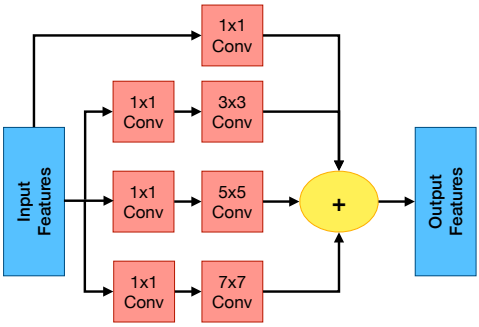

Figure 2. Inception block. Each $k \times k$ convolution with $k \geq 2$ is preceded by a 1×1 convolution to make the model efficient while having a deeper network. The final results are concatenated into the output feature. This is a slightly adjusted version of the original inception blocks [5] without a pooling operation as it was already used in [25].

3.1 Inception Blocks

We make use of the so-called inception blocks introduced in [5] for classification and object detection. Inception blocks introduce 1×1 convolutions before the higher dimensional convolutions (3×3, 5×5 in the original paper) to reduce the dimensionality of the feature maps. This densifies the representation and, as a result, reduces the computational cost because smaller feature maps cover the the same information. The needed computational cost, i.e. the number of multiplications, is reduced by an order of magnitude for $k \times k$ kernels with $k > 1$, and allows to use deeper networks without an efficiency loss. See Figure 2 for the structure of inception blocks. Using this, [5] managed to improve the state-of-the-art accuracy on ImageNet without a huge increase in complexity of the network.

3.2 Method

In our approach, we use the inception blocks in an encoder-decoder architecture to calculate depth maps from single images. The entire architecture can be seen in Figure 1. The inception structure leads to multi-level feature extraction in the image without increasing the complexity too much. This is the reason why we can achieve high accuracy with the least amount of parameters in our experiments (see Section 4.5). Additionally, inception blocks include cross-channel correlations independently of the spatial dimension in the 1×1 reduction. This is an advantage over many previous works where convolutions cover spatial and channel dimensions simultaneously without separating the information.

We also tested our architecture with the suggested auxiliary networks used with the inception blocks in [5]. However, we found that the network performs better without, and, additionally, has a lower number of parameters.

3.2.1 Network Architecture.

We optimized our network architecture to use as few parameters as possible without compromising the performance. As a result we use an adjusted inception block without max pooling (see Figure 2), and input the RGB image in half the resolution of the desired output depth resolution, as both did not have a significant influence on the results. Our final model architecture is comprised of 3 convolutional layers and 5 inception blocks in the encoder, and 4 inception blocks followed by 4 convolutions in the decoder, both with pooling operations in between. See Figure 1 for the full architecture.

Inception Blocks. We adjusted the original inception blocks from [5] slightly to fit this application better. Instead of convolutions up to kernel size 5×5, our inception block contains an additional convolution of size 7×7. The larger kernel size leads to more global feature information that helps with a consistent depth output. Each higher dimensional convolution is preceded by the inception block typical 1×1 convolution. We removed the max pooling of the inception block in [5] because it did not improve the results but increased the number of parameters. We show the structure of our inception block in Figure 2.

Convolutions. Additional to inception blocks, we used 3 basic convolutional layers in the lower stage of the encoder and 4 basic convolutional layers in the last stage of decoder part. Each basic convolutional layer is comprised of a convolution, a batch normalization, and a ReLU operation. Padding size of 1, 2, and 3 are used in 3×3, 5×5, 7×7 convolutions, respectively.

Pooling. Three max-pooling layers, with kernel size 2×2 and stride 2, are used in the encoder part to reduce the spatial dimension of features by preserving the most important feature information. In the decoder, 4 bi-linear up-sampling layers with scale factor 2 are used to upscale the dimension back to the image resolution for the final predicted depth.

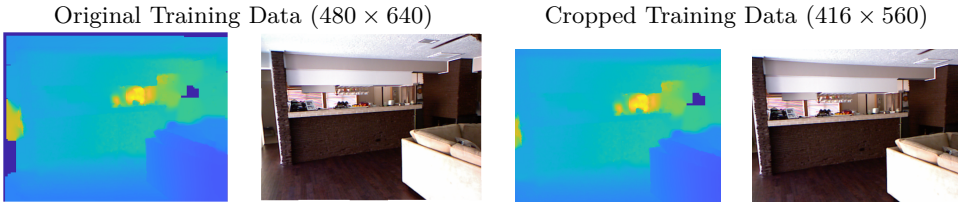

Figure 3. Comparison of original and cropped training data in the NYU dataset. The zero values along the boundaries vary within the training set and hinder efficient optimization.

4 Experiments

We show the performance of our architecture on the NYU and iBims datasets (Section 4.1) against several established monocular depth estimation methods. We compare the quantitative performance (Section 4.4) as well as the network size and training time (Section 4.5).

4.1 Datasets

We evaluate on the NYU v2 (Section 4.1.1) and iBims-1 (Section 4.1.2) datasets.

4.1.1 NYU v2 Dataset

In our experiments, we used the publicly available NYU Depth v2 RGB-D dataset [6]. This dataset contains 464 video sequences of 26 different indoor scenes recorded with a Microsoft Kinect RGB-D camera with over $400k$ frames in total. It contains two sets of 480×640 images, which are labeled (pre-processed) and raw sets. The labeled dataset accounts for 1449 pairs of RGB-D images which were randomly selected from the raw dataset. The second set is the raw dataset, which contains raw images (RGB and depth) and accelerometer dumps as produced by the Kinect camera. This set is unprocessed and with many missing depth pixels. Hence, we pre-processed and aligned the raw depth image using the acceleration data and tools provided with the dataset. The test images were randomly selected from the labeled images. The rest of the labeled images were added to training set. Then, the validation set was randomly selected from the whole training set.

We synchronized the RGB and depth images based on the timestamps of the captured frames. To increase the size and diversity in the training set, we augmented the training set with horizontally flipped versions which are meaningful for indoor scenes. The augmentation was done off-line as preprocessing. In total, we used $194K$ images (after augmentation) for training, 654 images for validation, and 654 images for testing. In our experiments comparing to existing works (Table 1), we have used the train-test split proposed by the dataset.

Cropped Training Data Due to the slight offset of depth and RGB frames and hardware, the training data contains varying amounts of missing boundaries. They

Figure 4. Output of our model on NYU RGB-D dataset. The first row shows the input RGB image, the second row shows the ground truth depth image, and the last row shows the estimated depth image of our method.

are quite irregular, and hindered our network from training efficiently. To overcome this, we cropped the training set at the boundaries to a smaller resolution, namely 416×560 pixels. See Figure 3 for a side-by-side comparison of the original and cropped versions. This improved our results and lowered the time and memory needed for training significantly. Notice that although we train on a lower resolution, our architecture is fully convolutional and, therefore, applicable to higher resolutions as well. We show the evaluation of both resolutions in Section 4.4.

4.1.2 iBims-1 Dataset

Additionally, we evaluate on the iBims-1 dataset [7]. This dataset contains 100 RGB-depth pairs from 10 different indoors scenes, and contains more in-depth ground-truth labeling and evaluation metrics than NYU. It is meant to further evaluate models trained on different datasets, especially NYU, showing generalization capability and providing more detailed metrics. All images are 480×640 pixels, with ground-truth depth, masks annotating invalid and missing pixels, as well as some semantic segmentation masks and plane estimations that are used for evaluation.

4.2 Implementation and Training

Our depth estimation network is implemented using PyTorch using Adam as the optimizer. The root mean squared error (RMSE) is used as the loss function. All models were trained with a batch size of 4.

The learning rate starts as 0.001 and is reduced by 10^{-1} as soon as the error on the validation set does not decrease anymore. This is also used as a stopping criterion. Table 3 shows the improvement of the RMS in each epoch with the corresponding learning rate. The training takes around 15 epoches and one epoch takes about 200 minutes. We choose the model with the best validation error as the final model. All experiments were done on a GeForce GTX GPU with 12 GB memory.

4.3 Evaluation Measures

We the following four standard metrics to compare the performance in our experiments. Here, y_i denotes the i^{th} pixel values in the ground-truth depth image y, $\hat{y}_i$ is the i^{th} pixel value in the predicted depth image y, and N is the total number of pixels of the depth image.

Root mean squared error: RMSE measures the normalized distance between the predicted value and the actual value. It is defined as follows:

$$\text{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Average relative error: RE is computed by dividing the absolute error by the magnitude of the ground-truth value:

$$\text{rel} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i}$$

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	rel $\downarrow$	rms $\downarrow$	$\log_{10} \downarrow$
Lee et al. [20]	0.885	0.978	0.994	0.110	0.392	0.047
Wei et al. [21]	0.875	0.976	0.994	0.108	0.416	0.048
Alhashim et al. [19]	0.895	0.980	0.996	**0.103**	0.390	**0.043**
Eigen et al. [11]	0.611	0.887	0.971	0.215	0.907	0.285
Xu et al. [14]	0.811	0.954	0.987	0.121	0.586	0.052
Laina et al. [17]	0.811	0.953	0.988	0.127	0.573	0.055
Hao et al. [13]	0.841	0.966	0.991	0.127	0.555	0.053
Li et al. [34]	0.788	0.958	0.991	0.143	0.635	0.063
Bhat et al. [35]	**0.903**	**0.984**	**0.997**	**0.103**	**0.364**	0.044
Chen et al. [25]	-	-	-	0.34	1.10	0.38
Ours (original resolution)	0.843	0.960	0.990	0.126	0.388	0.051
Ours (training resolution)	**0.937**	**0.989**	**0.997**	**0.076**	**0.253**	**0.031**

Table 1. Quantitative comparison on the NYU v2 RGB-D dataset. The first column shows the different methods for depth prediction. The next six rows show the standard metrics used to compare these methods. The up-arrow indicates that the largest value is the best, while the down-arrow indicates that the lowest value is the best. The bold values indicate the best model on that specific metric. The values for the other methods are taken from their respective papers. While we outperform on the resolution we trained on, we still get on-par accuracy on the original resolution that our network never saw during training.

Average $\log_{10}$ **error:** This is the mean of $\log_{10}$ scaled errors which weights down outliers in the solution:

$$\log_{10} = \frac{1}{N}\sum_{i=1}^{N}|\log_{10}(y_i) - \log_{10}(\hat{y}_i)|$$

Threshold accuracy: This measures the ratio of samples for which the relative error is below a threshold.

$$\delta_k = \frac{1}{N}\sum_{i=1}^{N}\sigma_i, \tag{1}$$

$$\sigma_i = \begin{cases} 1 & \text{if } \max(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}) < T^k, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In our experiments, we use $\delta_1, \delta_2, \delta_3$ and $T = 1.25$ for the evaluation. The first three measure the error, so values close to zero are better. Threshold accuracy is valued between 0 and 1 with 1 being the optimal value.

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	rel $\downarrow$	rms $\downarrow$	log $\downarrow$	$\epsilon_{\text{PE}}^{\text{plan}} \downarrow$	$\epsilon_{\text{PE}}^{\text{orie}} \downarrow$	$\epsilon_{\text{DBE}}^{\text{acc}} \downarrow$	$\epsilon_{\text{DBE}}^{\text{comp}} \downarrow$	$\epsilon_{\text{DDE}}^{+} \downarrow$	$\epsilon_{\text{DDE}}^{-} \downarrow$
Eigen [11]	0.36	0.65	0.84	0.32	1.55	0.17	6.65	25.62	5.48	70.31	25.71	**2.23**
Laina [17]	0.50	0.78	0.91	0.25	1.20	0.13	**5.71**	**18.49**	6.89	40.48	15.91	2.43
Li [34]	**0.59**	**0.85**	**0.95**	**0.22**	**1.07**	**0.11**	6.22	20.17	**3.68**	**36.27**	12.49	3.38
Ours	0.43	0.74	0.89	0.29	1.37	0.14	11.66	18.80	4.24	52.46	**2.14**	22.47

Table 2. Quantitative comparison on the iBims-1 dataset. The first six columns show the metrics as in Table 1, the next six are iBims specific, see Section 4.3. The up-arrow indicates that the largest value is the best, while the down-arrow indicates that the lowest value is the best. The bold values indicate the best model on that specific metric. The values for the other methods are taken from [7]. All of the methods were trained on NYU. Although the winner in the standard metrics is clear, the newly introduced metrics for iBims show varying results.

4.3.1 iBims Metrics

As an enrichment to the the standard metrics, the iBims dataset offers a collection of additional metrics based on the additional ground-truth annotation in the dataset. We will give a short overview of the measures here, please consult [7] for the exact formulas.

Planarity and Orientation: Based on ground-truth annotated major planes in the depth data, the values $\epsilon_{\text{PE}}^{\text{plan}}$ and $\epsilon_{\text{PE}}^{\text{orie}}$ measure how planar the result is in comparison to the ground-truth and how close the orientations are to each other, respectively.

Location Accuracy of Depth Boundaries: The next metric measures how well boundaries in the depth maps are preserved by comparing detected edges. $\epsilon_{\text{DBE}}^{\text{acc}}$ and $\epsilon_{\text{DBE}}^{\text{comp}}$ describe the accurarcy of existing edges as well as the completeness of present edges, respectively.

Directed Depth Error: $\epsilon_{\text{DDE}}^{+}$ and $\epsilon_{\text{DDE}}^{-}$ measure the percentages of pixels that were estimated as too far or too close, respectively.

4.4 Comparison with Existing Methods

We evaluate the quantitative performance of our methods against prior works in the same setting using the performance measures from Section 4.3.

4.4.1 NYU v2

We evaluate on both the original resolution of the NYU dataset as well our training resolution (see Section 4.1), using the same pretrained model for both. On the training resolution, our method performs extraordinarily well, outperforming the competitors in all measures (which are all normalized wrt. resolution). On the original resolution, our results are still close to state-of-the-art methods although this resolution was never seen during training. Notice that the NYU test set does not suffer from the same boundary issues the training set has. We are convinced that our architecture would perform even better on the original with a slightly adapted training procedure overcoming the missing boundary problems. The exact results

Epoch	Train RMS	Val RMS	LR
1	**0.889∗**	**0.604∗**	**0.001**
2	**0.509** ↓	**0.459** ↓	**0.001**
3	**0.390** ↓	**0.391** ↓	**0.001**
4	**0.333** ↓	**0.341** ↓	**0.001**
5	**0.299** ↓	**0.339** ↓	**0.001**
6	**0.277** ↓	**0.316** ↓	**0.001**
7	**0.261** ↓	**0.293** ↓	**0.001**
8	**0.248** ↓	**0.289** ↓	**0.001**
9	**0.239** ↓	**0.271** ↓	**0.001**
10	**0.231** ↓	**0.281** ↑	**0.001**
11	**0.207** ↓	**0.267** ↓	**0.0001**
12	**0.200** ↓	**0.256** ↓	**0.0001**
13	**0.197** ↓	**0.247** ↓	**0.0001**
14	**0.195** ↓	**0.259** ↑	**0.0001**
15	**0.193** ↓	**0.250** ↑	**0.00001**

Table 3. The RMSE on the training and validation set as well as the learning rate (LR) at each epoch. The learning rate is reduced by an order of magnitude each time the error on the validation set does not decrease anymore. The final stop happens after reducing the learning rate twice without improvement.

are reported in Table 1. Qualitative results can be seen in Figure 4.

4.4.2 iBims-1

On the iBims, dataset we compare to three methods in the benchmark of [7]. Since this dataset is rather new, not that many methods have been evaluated on it. All methods were pretrained on NYU and not finetuned to iBims. While the winner in the standard metrics is clear on this dataset, the results for the more detailed metrics of the dataset are mixed. Again, our method is not far off the state-of-the-art although our network is a lot smaller and was trained on a different resolution. The exact results are reported in Table 2. Qualitative results can be seen in Figure 5.

4.5 Number of Parameters

We compare our method in another metric, namely the number of trainable parameters in the architecture. This is not directly related to the quantitative performance but the number of parameters is still often critical for its performance and ability to generalize to new inputs. While more parameters can be more expressive, they also need longer for training and more training data to avoid overfitting. In general, a smaller network with the same performance on the a task is preferable.

Our model has $16.7M$ trainable parameters which is less than all except one competitors, and a third of the next larger architecture. See Table 5 for the details of our architecture. The only network smaller than ours is from [25]. However, as can

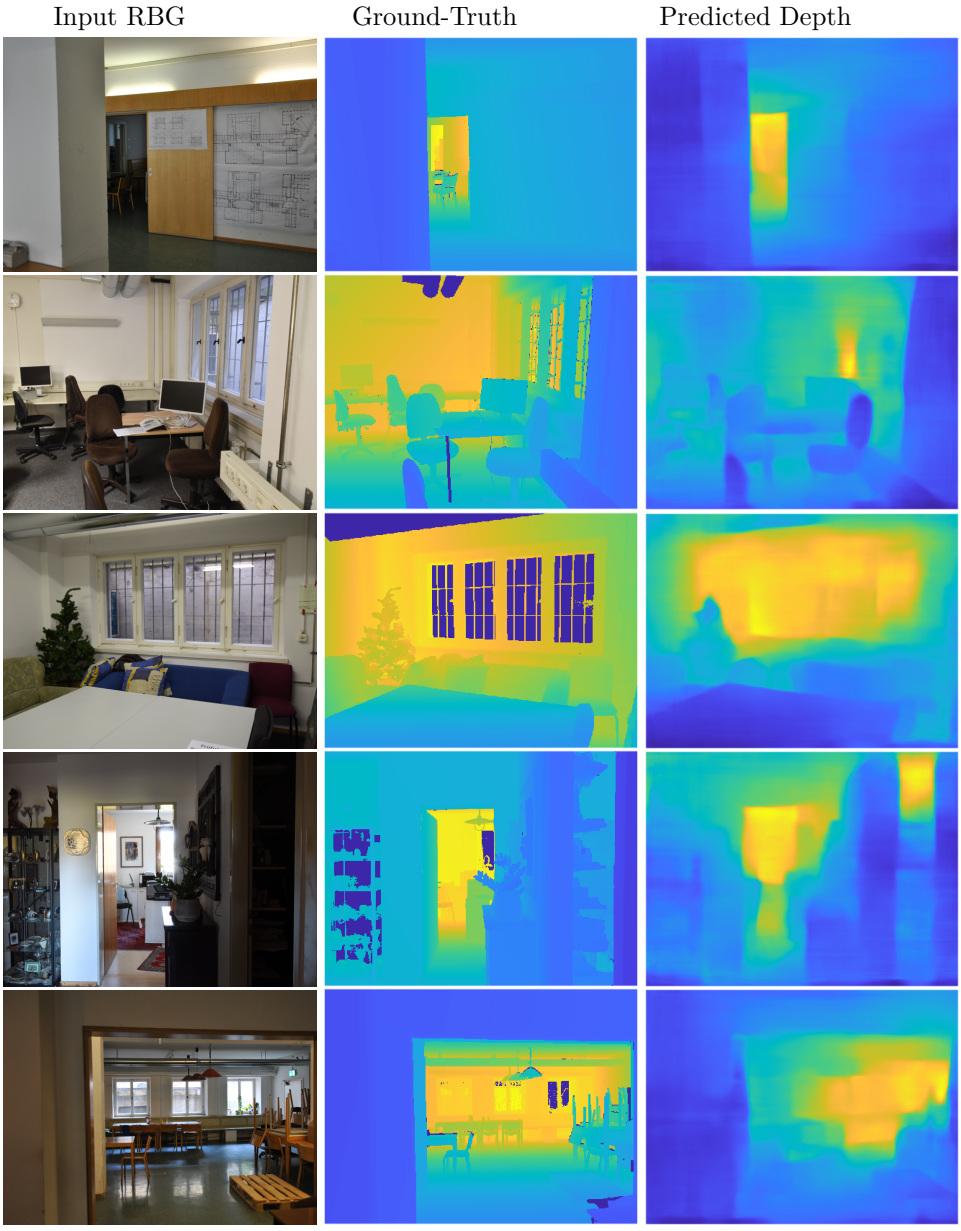

Figure 5. Output of our model on iBims-1 dataset. The first row shows the input RGB image (cropped), the second row shows the ground truth depth image, and the last row shows the estimated depth image of our method. Notice that our model was neither trained on this resolution nor the same dataset.

Method	**# of Learnable Parameters**
Lee et al. [20]	$47.0M$
Wei et al. [21]	$90.4M$
Alhashim et al. [19]	$42.6M$
Eigen et al. [11]	$240.8M$
Laina et al. [17]	$63.5M$
Fu et al. [15]	$110.0M$
Bhat et al. [35]	$78.2M$
Chen et al. [25]	$5.3M$
Proposed model	$\mathbf{16.7M}$

Table 4. Comparison on the number of learnable parameters. The numbers are in million. Although our method has the least number of parameters by a large margin, we outperform all other methods in the quantitative evaluations in Table 1.

be seen in Table 1, it falls behind in terms of accuracy in all given measures. The

Layer	Kernel size	Output features	1 × 1 out	3 × 3 red out	3 × 3 out	5 × 5 red out	5 × 5 out	7 × 7 red out	7 × 7 out
Encoder Network									
Basic Conv1	7	32	-	-	-	-	-	-	-
Basic Conv2	5	64	-	-	-	-	-	-	-
Basic Conv3	3	128	-	-	-	-	-	-	-
Max Pool1	2	128	-	-	-	-	-	-	-
Inception1	-	384	64	96	192	32	64	32	64
Inception2	-	672	192	112	256	32	96	64	128
Max Pool2	2	672	-	-	-	-	-	-	-
Inception3	-	760	224	156	312	64	128	32	96
Inception4	-	1280	384	192	384	92	256	64	256
Max Pool3	2	672	-	-	-	-	-	-	-
Inception5	-	2048	768	256	768	96	256	96	256
Decoder Network									
Inception1	-	1280	512	192	384	32	256	64	128
Up Pool1	2	1280	-	-	-	-	-	-	-
Inception2	-	776	224	164	328	64	128	32	96
Inception3	-	480	128	96	224	32	64	32	64
Up Pool2	2	480	-	-	-	-	-	-	-
Inception4	-	256	64	96	128	16	32	16	32
Basic Conv1	3	128	-	-	-	-	-	-	-
Up Pool3	2	128	-	-	-	-	-	-	-
Basic Conv2	3	32	-	-	-	-	-	-	-
Up Pool4	2	32	-	-	-	-	-	-	-
Basic Conv3	3	16	-	-	-	-	-	-	-
Basic Conv4	3	1	-	-	-	-	-	-	-

Table 5. Detail of our model architecture. $n \times n$ *out* is the number of output channels after a convolution by $n \times n$, $n \times n$ *red out* is the number of output channels after 1×1 convolution, which is applied before $n \times n$ convolution, where n is the kernel size.

number of parameters of all methods can be found in Table 4. Additionally, and as a result of this, our model also needs fewer iterations before training converges. We train our method for $218.1k$ iterations compared to [19] and [15] which need $1M$, $3M$ iterations, respectively. The number of training iterations is directly related to the power consumption needed, and therefore less iterations are preferable.

5 Conclusion

We presented a novel encoder-decoder architecture with modified inception blocks for the challenging task of monocular depth estimation. The inception blocks focus on finding cross-channel relationships instead of relying on the spatial information of convolutions only or including geometric priors in the architecture as state-of-the-art methods do. We are able to reach close to state-of-the-art results on the NYU v2 dataset while using a considerably lower amount of trainable network parameters. We also showed how our network is generalizable to different resolutions and datasets by training on a smaller resolution on NYU, and evaluated the pretrained NYU model on the iBims dataset. Our method only needs a third of the parameters of the next larger competitor with similar accuracy, and can be trained in less iterations. In addition to speed, the training time is directly correlated to the electricity needed, a measure that cannot be disregarded in light of recent developments of the climate and environment. In future work we plan to directly measure the energy consumption of our methods instead of relying on indirect measurement through training time.

6 Acknowledgments

The project was conducted with the support of the Erasmus+ programme of the European Union. Emanuele Rodolà is supported by the ERC grant no. 802554 (SPECGEO).

7 References

[1] P. Kamencay, M. Breznan, R. Jarina, P. Lukac, and M. Zachariasova, "Improved depth map estimation from stereo images based on hybrid method.," *Radioengineering*, vol. 21, no. 1, 2012.

[2] W. Zhou, E. Zhou, Y. Yan, L. Lin, and A. Lumsdaine, "Learning depth cues from focal stack for light field depth estimation," in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, pp. 1074–1078.

[3] C. Hazirbas, S. G. Soyer, M. C. Staab, L. Leal-Taixé, and D. Cremers, "Deep depth from focus," in *Asian Conference on Computer Vision (ACCV)*, 2018.

[4] H. Tang, S. Cohen, B. Price, S. Schiller, and K. N. Kutulakos, "Depth from defocus in the wild," in *IEEE International Conference on Computer Vision (CVPR)*, 2017.

[5] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[6] P. K. Nathan Silberman Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision (ECCV)*, 2012.

[7] T. Koch, L. Liebel, F. Fraundorfer, and M. Körner, "Evaluation of cnn-based single-image depth estimation methods," in *Proceedings of European Conference on Computer Vision Workshops*, 2018, pp. 31–348.

[8] C.-Q. Zhao, Q.-Y. Sun, C.-Z. Zhang, Y. Tang, and F. Quian, "Monocular depth estimation based on deep learning: An overview," *Science China Technological Sciences*, vol. 63, 2020.

[9] H. Laga, L. V. Jospin, F. Boussaid, and M. Bennamoun, "A survey on deep learning techniques for stereo-based depth estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[10] R. Nair, K. Ruhl, F. Lenzen, *et al.*, "Time-of-flight and depth imaging. sensors, algorithms, and applications.," *Lecture Notes in Computer Science*, vol. 8200, 2013.

[11] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[12] H. Ren, M. El-Khamy, and J. Lee, "Deep robust single image depth estimation neural network using scene understanding.," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 37–45.

[13] Z. Hao, Y. Li, S. You, and F. Lu, "Detail preserving depth estimation from a single image using attention guided networks," in *2018 International Conference on 3D Vision (3DV)*, IEEE, 2018, pp. 304–313.

[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 the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5354–5362.

[15] 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*, 2018, pp. 2002–2011.

[16] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 6101–6108.

[17] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*, IEEE, 2016, pp. 239–248.

[18] S. Gur and L. Wolf, "Single image depth estimation trained via depth from defocus cues," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7683–7692.

[19] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv preprint arXiv:1812.11941*, 2018.

[20] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[21] Y. Wei, Y. Liu, C. Shen, and Y. Yan, "Enforcing geometric constraints of virtual normal for depth prediction," *arXiv preprint arXiv:1907.12209*, 2019.

[22] Y. Chen, H. Zhao, and Z. Hu, "Attention-based context aggregation network for monocular depth estimation," *arXiv preprint arXiv:1901.10137*, 2019.

[23] 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 (CVPR)*, 2016, pp. 770–778.

[24] M. Ramamonjisoa and V. Lepetit, "Sharpnet: Fast and accurate recovery of occluding contours in monocular depth estimation," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[25] W. Chen, Z. Fu, D. Yang, and J. Deng, in *Advances in Neural Information Processing Systems (NIPS*.

[26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision (ECCV)*, 2020.

[27] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, 2017.

[28] A. G. Howard, M. Zhu, B. Chen, *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[31] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, 2019.

[32] X. Tu, C. Xu, S. Liu, *et al.*, "Efficient monocular depth estimation for edge devices in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, 2021.

[33] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 97, 2019.

[34] J. Li, R. Klein, and A. Yao, "A two-streamed network for estimating fine-scaled depth maps from single rgb images," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[35] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.