Comparing Unsupervised Depth Estimation Modeling and Alternative Methods

Alex Evelson
Stanford University
aevelson@stanford.edu

Parker Killion
Stanford University
pkillion@stanford.edu

Danny Schwartz
Stanford University
deschwa2@stanford.edu

Abstract

Depth estimation has a variety of important applications in the development of autonomous robots, ranging from those used in the manufacturing sector to commercial self-driving vehicles. Although traditional computer vision techniques like triangulation can often accurately estimate the depth in a scene with stereo camera data, monocular approaches often suffer a loss of accuracy in comparison. Recently, Godard and Aodha published research about a method to train a monocular neural depth estimator without expensive-to-obtain depth labels [2].

To study the practicality of this approach, we train and evaluate a series of machine learning models for the depth estimation task. We apply various depth estimation techniques to a dataset often used in autonomous driving research and perform an analysis of their performance.

1. Introduction

Depth estimation is important in many industrial robotics and computer vision tasks. With the recent surge in popularity of deep neural networks, monocular neural depth estimators have been trained and deployed to this end. Monocular estimation is preferred in some settings because stereo camera systems are often more expensive and complex to install and operate. To explore the various approaches to depth estimation, we compare a traditional computer vision approach and a series of machine learning models.

In our approach to the problem of depth estimation, we use the KITTI Depth Prediction dataset ([8]), a well-established source with a combination of rectified RGB images and depth maps, which we consider our ground truths.

As a baseline, we evaluate a stereo block-matching algorithm implemented in OpenCV ([6]). We also develop our own neural network models. Specifically, we train and assess four supervised depth prediction models, two unsupervised depth prediction models, and two semi-supervised depth prediction models. All of our models use the same encoder-decoder architecture, with only the first layer of the stereo models changing to accommodate twice the number of input channels. Half of our models are trained with a monocular setting, and the rest are trained with a stereo setting.

For our unsupervised models, we use the technique detailed in [2]. We train supervised models using mean squared error and scale-invariant-log error as optimization objectives.

Our semi-supervised models combine these approaches. We provide sample depth predictions and evaluate all models based on two relevant metrics. All of our algorithms output disparity maps, which we then convert to depth predictions. Qualitatively, we visually compare the generated depth maps of our various methods to one another and to the ground truth depth values provided in our dataset. We also use our generated disparity maps to reconstruct the left images based on the right images. Quantitatively, we calculate the error between the resulting depth maps and the ground truth using the mean squared error and scale invariant logarithmic error metrics.

Given the significant ambiguity associated with removing one camera from the pair, we expect stereo techniques to outperform monocular techniques in general, but we expect the monocular models to perform well enough to reasonably estimate the depths in the image.

2. Related Work

This is not the first project to evaluate a range of pre-existing techniques for depth estimation. In 2019, Amlaan Bhoi conducted a review of existing papers exploring various machine learning based techniques for monocular depth estimation ([1]). Whereas Bhoi's paper is limited to comparing existing papers and only focuses on monocular depth estimation, we implement a series of approaches ourselves and look beyond monocular depth estimation to consider the performance of stereo models and traditional computer vision techniques. Like [1], this paper includes the performance of supervised, semi-supervised, and unsupervised neural models.

Practical depth estimators are not always learned statistical models; *a priori* algorithms like block-matching (introduced in [5]) are also used in industrial applications.

2.1. Unsupervised Monocular Depth Estimation with Left-Right Consistency [2]

The main work our project is based on proposes an optimization objective for a neural depth estimator that does not require depth labels. A diagram of the approach to loss computation is shown in figure 2. The paper describes an encoder-decoder neural model that takes as input the left image of a rectified stereo pair and produces various predicted disparity maps. Pairs of left-to-right and right-to-left disparity maps are produced at full resolution, half-resolution, quarter-resolution, and eighth-resolution. The objective function is a sum of terms collected from each of these disparity pairs, and for each particular disparity pair, three distinct loss terms are used for each image in the pair.

Two of these loss terms use a differentiable sampling operation also used in [4].

The first of these loss terms is the image reconstruction term (see equation 1). A reconstructed image $\overline{I}$ (obtained by applying the sampling operation from [4] to the opposite image using the corresponding predicted disparity map) is compared to the original image I with an element-wise L1 norm and the structural similarity (SSIM) metric and the mean of these per-pixel scores is taken over all pixels.

The second of these loss terms is the disparity smoothness term (see equation 2). The approximate spatial gradients of disparity incur a penalty if the spatial gradients of the pixel color at that same location are low, enforcing smooth disparities in visually smooth regions.

The third and final loss term is the titular left-right consistency term (see equation 3). Using a differentiable sampler, one disparity in the pair is compared to the other disparity in the pair differentiably sampled using the first disparity. These two disparities are expected to be approximately equal given the relationship between the real disparity values for any stereo-captured scene, so the model is penalized if there is a difference.

$$C^l_{ap} = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - \text{SSIM}(I^l_{ij}, \overline{I}^l_{ij})}{2} + (1-\alpha)||I^l_{ij} - \overline{I}^l_{ij}|| \tag{1}$$

The image reconstruction loss term for the left image in the stereo pair.

$$C^l_{ds} = \frac{1}{N}\sum_{i,j} |\partial_x d^l_{ij}| e^{-||\partial_x I^l_{ij}||} + |\partial_y d^l_{ij}| e^{-||\partial_y I^l_{ij}||} \tag{2}$$

The disparity smoothness loss term for the left image in the stereo pair.

$$C^l_{lr} = \frac{1}{N}\sum_{i,j} |d^l_{ij} - d^r_{ij+d^l_{ij}}| \tag{3}$$

The left-right consistency loss term for the left image in the stereo pair.

The authors of this work trained a model using this objective on a KITTI dataset and found the resulting depth estimates to outperform some contemporary supervised approaches.

3. Approach

3.1. Methods

We apply a range of classical and machine learning based approaches to depth estimation, the results of which we are then able to compare.

3.1.1 Stereo Block-Matching

For the traditional approach in a stereo system, we use a stereo block-matching algorithm (based on [5]) implemented in OpenCV ([6]). We expected this approach to compare favorably to monocular models because of the advantage it has in using stereo inputs. The approach relies on two parameters: the number of disparities and the block size. The number of disparities defines the number of pixel positions the algorithm evaluates as disparity candidates. The block size dictates the size of the blocks compared by the algorithm in each of these evaluations. We use an odd block size so that the candidate pixel positions are at the centers of each evaluated block. The larger the block size, the smoother, but less accurate, the predicted disparity map.

This algorithm is known to struggle in regions without much texture. Under certain conditions, the algorithm will choose not to make a disparity prediction for a pixel. This means that, unlike our neural disparity models, the output of the stereo-block matching algorithm is sparse. This sparsity presents a challenge to evaluation that is addressed in 5.2.1.

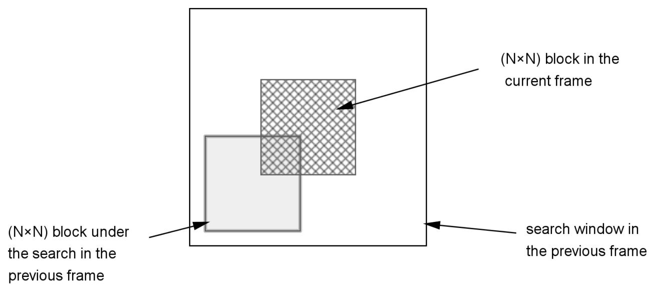

(N×N) block in the current frame
(N×N) block under the search in the previous frame
search window in the previous frame

Figure 1. Visualization of the Stereo Block-Matching Algorithm

3.1.2 Neural Network Architecture

All of our neural models use the same network architecture. As in [2], our neural model has four output scales and skip connections between equivalently-sized layers that bypass the bottleneck in the CNN. In the supervised-only settings, the intermediate-resolution outputs are ignored. The specific encoder-decoder architecture we use is detailed in table 2.

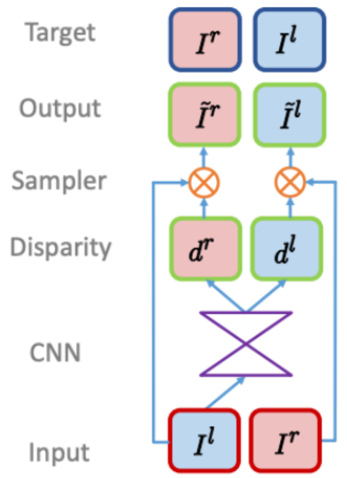


Figure 2. Unsupervised Model's Training Configuration [2]

Though the original paper was accompanied by a Tensorflow implementation ([3]), we chose to re-implement any code related to model training and evaluation in PyTorch. This includes the multi-scale loss function defined in [2]. Our loss function is a slight variation of the original; we use a 3x3 Gaussian kernel for SSIM instead of the 3x3 block kernel used for SSIM in [2]. We also use Sobel filters to obtain spatial image gradient approximations instead of just the raw difference between adjacent elements used in the original work. Outside of these changes, our loss function is the same, even down to the choice of hyperparameters. This includes the image reconstruction loss, the disparity smoothness loss, and the left-right consistency loss terms discussed in section 2.1. As in [2], the final activation layer on the model is a sigmoid activation scaled by 0.3. Lower-resolution disparity map predictions are extracted from the first two channels of certain layer outputs (just before activation functions are applied). The final activation function 0.3 * sigmoid is applied instead to obtain intermediate-resolution disparity maps.

3.1.3 Stereo Models

The only change we make to our model for the stereo cases is that we pass in pairs of images into the model instead of using just the left images. We accomplish this by merely adjusting the expected number of channels at the model's first stage and stacking the two images in the channel dimension.

3.1.4 Supervised Models

We also train supervised models. For each one, we use a loss function to compare the model's predicted depth with the ground-truth depth labels provided in the dataset. For one of the models, we use mean squared error as the loss function. For the other, we use scale-invariant log error.

Mean Squared Error Mean squared error is a simple, commonly-used regression objective that we apply as a loss function in supervised learning. Since we expect that errors tend to be greater at pixels whose actual depth is greater, this metric is more heavily influenced by pixels that are farther away from the camera, which might cause undesired model behavior.

Mean squared error is defined as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{groundTruth}_i - \text{predicted}_i)^2 \tag{4}$$

Scale Invariant Logarithmic Error Scale invariant logarithmic error (SILog) is another performance metric used to assess depth prediction quality on the KITTI dataset. Scale-invariance helps avoid the error term being dominated by errors in pixels depicting far-away objects, but it is less intuitive to interpret.

SILog is defined as follows:

$$\begin{aligned}\text{SILog} = &\frac{1}{n}\sum_{i=1}^{n}(\text{groundTruth}_i - \text{predicted}_i)^2 \\ &-(\frac{1}{n}\sum_{i=1}^{n}\text{groundTruth}_i - \text{predicted}_i)^2\end{aligned} \tag{5}$$

3.1.5 Semi-Supervised Models

We perform experiments where we combined the unsupervised objective with the mean squared error objective via simple summation. In these settings, a supervision signal is only taken from images recorded on September 28, 2011 – about 1.14% of the images in the dataset.

4. Experiments

4.1. Dataset

We use the KITTI Depth Prediction dataset ([8]). KITTI consists of images taken from cameras mounted on cars and labeled by human annotators, so all of these images are of settings like highways, streets, and intersections. Specifically, the Depth Prediction dataset contains 61 image sequences of 375x1242-pixel rectified RGB image pairs and depth maps – a total of 42,378 images. The depth maps, generated using LiDAR scans, are relatively sparse; in a sample of depth maps from our dataset, only about 4% of the pixels actually had depth labels.

The KITTI data is split over 5 different dates with each having slightly different associated camera calibration data. We split the data into training, validation, and test sets, making sure that images from one sequence were bucketed into exactly one of these sets so as not to produce overly-optimistic test results. Our data split is 93.4% for training, 3.69% for validation, and 2.87% for testing.

When creating our own dataloader, we reused some logic from Monodepth2 ([7]), a Github project that uses PyTorch to train depth models on this dataset.

4.2. Methods

We prepare and evaluate several algorithms to predict depth for the images in our dataset.

All neural models were trained with the Adam optimizer using the same optimizer hyperparameters as in [2], but using a slowly decaying learning rate schedule instead of the constant learning rate used in [2]. In each setting, we selected the model parameters from the epoch in which we recorded the lowest validation loss.

Our neural models were trained on Google Cloud Platform virtual machines equipped with NVIDIA Tesla K80 GPUs.

4.2.1 Stereo Block-Matching

For our stereo block-matching strategy, we use a block size of 15 pixels and create a disparity map using 368 disparities. We chose this number of disparities because it allows the block-matching algorithm to make predictions in roughly the same range of disparities the neural models can.

4.2.2 Supervised Models

We trained each supervised model for 15 epochs, a process which took approximately 15 hours per model.

4.2.3 Unsupervised Models

We trained our unsupervised models for 18 epochs each. We found the convergence of the monocular unsupervised model to be fairly unstable, so for the first 14 training epochs we up-weighted the image reconstruction loss term by a factor of 100 and then for the remaining 4 training epochs we up-weighted the disparity smoothness loss terms by a factor of 10. We find this produces better results than using the hyperparameters from [2]. Training these models took about two days each.

4.2.4 Semi-Supervised Models

To avoid the immense amount of training time required to train the semi-supervised models from randomly initialized weights, we initialized the weights of the semi-supervised models we trained to be the weights of the corresponding unsupervised models we had previously trained. We then further trained these models on the combined semi-supervised objective for 6 epochs each.

We noticed during training that, for a number of epochs, the unsupervised loss decreases while the supervised loss decreases until they both eventually decrease in tandem. This period was substantially longer in the stereo setting, to the extent where we believe training longer may have significantly improved the semi-supervised stereo model's performance.

5. Results and Evaluation

To evaluate our results, we are using a mix of qualitative and quantitative metrics. Once we obtain the disparity maps from each of our implemented methods, we convert them into depth maps by applying equation 6, multiplying the inverted disparity values by the baseline and focal length of the camera used to obtain the initial images.

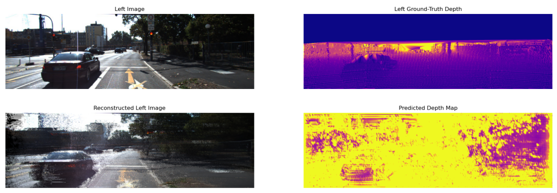
$$\text{depth} = \frac{\text{focal length} \times \text{baseline}}{\text{disparity}} \tag{6}$$

5.1. Qualitative Results and Evaluation

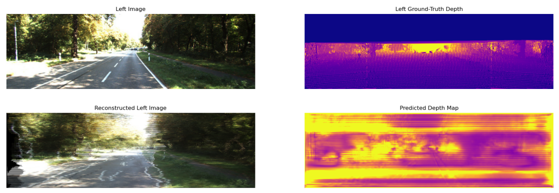
We present selected output samples for our neural models in figure 3 and for our stereo block-matching algorithm in figure 4. We provide the predicted depth maps, the sparse ground truths present in the KITTI dataset and, for the neural models, recreate the left images from their stereo counterparts based on our calculated disparities. Because the ground truth depth maps are sparse, we interpolate them to aid in the visualization process. We use the same color scheme across all depth maps: yellow signifies a larger depth and blue signifies a smaller depth.

As the results demonstrate, the stereo block-matching algorithm does a significantly better job outputting a depth map that visually reflects the difference between images across all depths. Compared to the monocular models, this is not particularly surprising considering that the block-matching algorithm benefits from access to both images in the stereo pair. The stereo block-matching algorithm's depth prediction here is also free from the kinds of noise seen in those of the neural models. We only trained our neural models for between 15 and 24 epochs, whereas in [2], the authors used 50 epochs. We suspect that training for longer may have improved the visual performance of our neural models relative to the stereo block-matching approach.

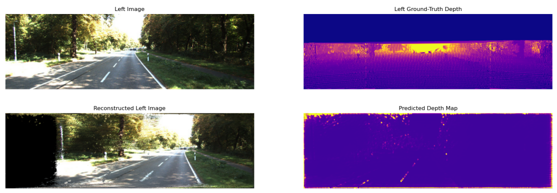
Visually, it appears that both the supervised and unsupervised models have a tendency to overestimate the depth of objects in the image, especially when using SILog as a loss function. However, certain models are still able to produce predicted depth maps in which the various depths of



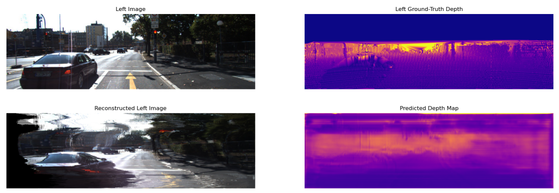
Unsupervised Monocular



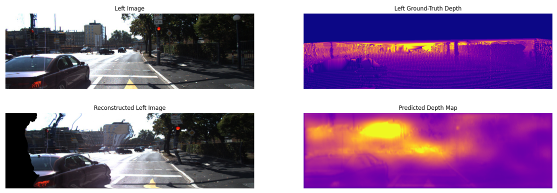
Unsupervised Stereo



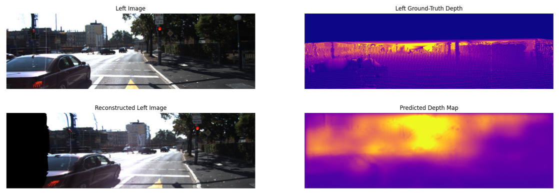
Semi-Supervised Monocular



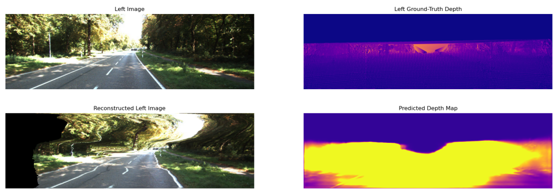
Semi-Supervised Stereo



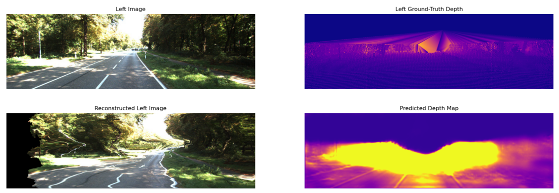
Supervised Monocular (MSE)



Supervised Stereo (MSE)



Supervised Monocular (SILog)



Supervised Stereo (SILog)

Figure 3. Neural Model Qualitative Results

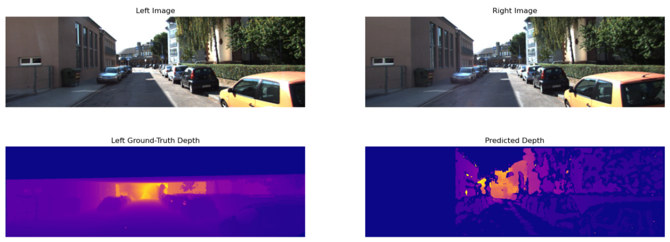


Figure 4. Qualitative Comparison of the Stereo Block-Matching Algorithm on Selected Image

the objects in the image can be visually discerned. For example, the unsupervised monocular model, semi-supervised stereo model, and both supervised models with an MSE objective function estimate relative depth accurately enough for a viewer to be able to discern the car in the image's foreground using the generated depth map. Similarly, the supervised stereo model produces an image in which a viewer can see the road getting progressively further away.

Interestingly, the unsupervised monocular model appears to overestimate the depth of points in the scene while the semi-supervised monocular model appears to underestimate the depth of points in the scene. It could be that adding supervision helps shift what the model is effectively using as the mean depth in many cases. The bright yellow regions in the unsupervised stereo setting are also much more muted in the semi-supervised stereo example. We believe this demonstrates an advantage of adding some supervision signal to this still largely unsupervised objective.

Depth Estimation Algorithm	MSE	SILog
Stereo Block-Matching	42.81	2.02
Supervised (MSE) Monocular Model	23.12	2.19
Supervised (MSE) Stereo Model	22.91	2.17
Supervised (SILog) Monocular Model	23.37	6.08
Supervised (SILog) Stereo Model	23.31	4.32
Unsupervised Monocular Model	23.28	5.43
Unsupervised Stereo Model	23.03	1.87
Semi-Supervised Monocular Model	**22.11**	**1.55**
Semi-Supervised Stereo Model	22.92	2.29

Table 1. Quantitative Results

5.2. Quantitative Results and Evaluation

Beyond looking at the images to get a qualitative sense of our results, we also use several quantitative metrics to score each of our approaches. Namely, we use a modified version of mean squared error (MSE) and scale invariant logarithmic error.

Table 1 shows the results we obtained for each of our approaches evaluated against our test set.

5.2.1 Metric Adjustments

For the outputs of the neural models, we use a slightly modified version of each quantitative evaluation metric to account for the fact that our ground truth depth maps contain a wide range of values. Namely, we only consider certain values in the ground truth depth maps. Because they are sparse, we set a minimum depth value of 0.05m to ignore the 0 values. Furthermore, they contain some values that are so high that they would dominate the rest of our results in the computation of error. We therefore set a maximum ground-truth depth value of 20m. We then only calculate our error metrics based on those pixel values where the ground truth depth map falls within our range, and our n (as defined in equations 4 and 5) is thus only the number of pixels used for the computation.

We similarly alter our error metrics for the stereo block-matching strategy, though we need to make additional adjustments here. The block-matching algorithm returns a special negative disparity value for the pixels for which it fails to find a match, so we must further filter out those points when calculating the error metrics and adjust n accordingly.

5.2.2 Interpretation of Quantitative Results

Surprisingly, the stereo-block matching algorithm performs quite poorly for the metrics we selected despite having the depth map with the cleanest visual quality. This may be because the modeling assumptions associated with stereo block-matching aren't as realistic as those used in the unsupervised training objective, and it might compare unfavorably to the supervised models we evaluated because they have access to the ground truth in their training objective while block matching is entirely *a priori*.

Another surprise was that the supervised models explicitly trained to optimize SILog performed worse on SILog than almost all of the other models. They also have a slightly higher MSE score than the other models. We conclude that SILog is not a good choice of optimization objective for this task.

In general, it appears that the unsupervised models did quite well compared to their supervised counterparts, and that the stereo models outperformed the monocular models. There is an inconsistency here though; the semi-supervised stereo model performed considerably worse than the semi-supervised monocular model.

We discuss the phenomena observed in our semi-supervised training procedure in section 4.2.4; our conclusion is that the semi-supervised stereo model may have performed significantly better if trained for longer.

6. Conclusion

After a thorough study of several varieties of models, we have reinforced that the unsupervised objective in [2] performs well on real-world data. We did find the objective to be somewhat unstable, but this was greatly helped by the addition of a relatively small fraction of labeled examples and a corresponding supervised objective term. We found this to work well in the setting where ground-truth depth labels were sparse, so it should be flexible enough to help industrial practitioners enhance their unsupervised depth models with however much supervision data they can manage to add.

It would be interesting to vary the fraction of data that a supervision objective is applied for to analyze the effects on performance and convergence speed of the models. It could also be quite interesting to use sparse depth prediction heuristics to enhance the objective by nudging it toward depth values the practitioner is confident enough to provide.

Ultimately, we believe the unsupervised objective in [2] has great practical potential and there is opportunity to combine it with other depth prediction strategies.

References

[1] A. Bhoi. Monocular depth estimation: A survey. 2019. 1

[2] G. J. B. Clément Godard, Oisin Mac Aodha. Unsupervised monocular depth estimation with left-right consistency. *Conference on Computer Vision and Pattern Recognition*, pages 234–778, 2017. 1, 2, 3, 4, 6

[3] C. Godard, D. Turmukhambetov, Z. Gang, and G. Osipov. monodepth github repo, 2017. 3

[4] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015. 2

[5] K. Konolige. Small vision systems: Hardware and implementation. In Y. Shirai and S. Hirose, editors, *Robotics Research*, pages 203–212, London, 1998. Springer London. 1, 2

[6] OpenCV. Opencv: Depth map from stereo images. `https://docs.opencv.org/3.4/dd/d53/tutorial_py_depthmap.html`. 1, 2

[7] D. T. Tristan Rice, Clement Godard. Monodepth2 a pytorch implementation, 2019. 4

[8] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. `http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_prediction`, 2017. 1, 3

Supplementary Materials

A link to our GitHub Repository:
https://github.com/danny-schwartz7/monodepth2

Index	Layer Type	Filter Size	Stride	# Output Channels	Dimensions After Upsampling	Skip Conn Index	Nonlinearity
0	Conv2D	(5, 5)	(2, 2)	4	N/A	N/A	ELU
1	Conv2D	(5, 5)	(2, 2)	4	N/A	N/A	ELU
2	Conv2D	(5, 5)	(2, 2)	8	N/A	N/A	ELU
3	Conv2D	(3, 7)	(2, 3)	16	N/A	N/A	ELU
4	Conv2D	(3, 7)	(2, 2)	32	N/A	N/A	ELU
5	Conv2D	(3, 7)	(1, 1)	4	N/A	N/A	ELU
6	Flatten	N/A	N/A	512	N/A	N/A	None
7	Linear	N/A	N/A	128	N/A	N/A	None
8	BatchNorm	N/A	N/A	128	N/A	N/A	ELU
9	Linear	N/A	N/A	128	N/A	N/A	None
10	BatchNorm	N/A	N/A	128	N/A	N/A	ELU
11	Reshape to (8, 16)	N/A	N/A	1	N/A	N/A	None
12	ConvTranspose2D	(1, 1)	(1, 1)	4	N/A	N/A	ELU
13	BilinearUpsample	N/A	N/A	4	(10, 22)	N/A	None
14	Conv2D	(3, 3)	(1, 1)	32	N/A	N/A	ELU
15	BilinearUpsample	N/A	N/A	32	(21, 49)	N/A	None
16	Conv2D	(3, 3)	(1, 1)	16	N/A	N/A	ELU
17	BilinearUpsample	N/A	N/A	32	(21, 49)	N/A	None
18	Conv2D	(3, 3)	(1, 1)	16	N/A	N/A	ELU
19	BilinearUpsample	N/A	N/A	16	(44, 152)	N/A	None
20	Conv2D	(5, 5)	(1, 1)	8	N/A	N/A	ELU
21	BilinearUpsample	N/A	N/A	8	(91, 308)	3	None
22	Conv2D	(5, 5)	(1, 1)	4	N/A	N/A	ELU
23	BilinearUpsample	N/A	N/A	4	(186, 619)	2	None
24	Conv2D	(5, 5)	(1, 1)	4	N/A	N/A	ELU
25	BilinearUpsample	N/A	N/A	4	(375, 1242)	1	None
26	Conv2D	(5, 5)	(1, 1)	4	N/A	N/A	ELU
27	Conv2D	(7, 7)	(1, 1)	4	N/A	N/A	ELU
28	Conv2D	(7, 7)	(1, 1)	4	N/A	N/A	ELU
29	Conv2D	(7, 7)	(1, 1)	2	N/A	26	0.3 * sigmoid

Table 2. Encoder-Decoder architecture used for unsupervised monocular depth model. Skip connections are specified such that the output of the specified output layer is added to the input of the specified input layer before the layer operation is conducted. The first two channels of the output of layers 0, 1, and 2 are taken before applying the ELU nonlinearity and 0.3 * sigmoid is applied instead to obtain left and right depth maps at the corresponding lower resolutions. Lower-resolution disparity map predictions are extracted from the first two channels of the outputs just before the activations of layers 24, 22, and 20. The activation function 0.3 * sigmoid is applied to obtain disparity maps.