Self-Supervised Monocular Depth Estimation with Uncertainty

Minjune Hwang[*]
Department of Computer Science
Stanford University
mjhwang@stanford.edu

Adithya Ganesh[*]
Department of Computer Science
Stanford University
acganesh@stanford.edu

Abstract

Depth estimation is an essential component of various systems, with applications in scene understanding, autonomous driving, and medical imaging. Given the safety critical nature of many of these tasks, we ask the following question: can we develop a model that can produce per-pixel depth estimates as well as per-pixel measures of uncertainty? In this work, we present three methods that produce depth estimates with uncertainty: ENSEMBLEDEPTH, DROPOUTDEPTH, *and* AUGMENTDEPTH. *Overall, we found that all three methods were competitive in RMSE with the model trained in Monodepth2[13] (RMSE 4.863), e.g.* ENSEMBLEDEPTH *had an RMSE as low as 4.666, outperforming this baseline. We examine model uncertainty outputs qualitatively and quantitatively, and notice that larger values of model uncertainty often correlate with model errors or qualitatively difficult sections of a scene.* DROPOUTDEPTH *in particular had the largest correlation between per-pixel model uncertainty and RMSE model error.*

1. Introduction

Depth estimation is a task in which we infer a dense depth map from one or more input images. As depth information conveys fundamental information about the 3D structure of scene, it is widely used in various tasks including autonomous driving, scene reconstruction, motion estimation, augmented reality, and medical imaging. While conventional methods utilize devices like RGB-D cameras, radar, or LIDAR to estimate depth information, recent studies aim to extract depth information from simple 2D images.

Collecting ground truth depth data for training supervised learning models can be non-trivial and expensive. As such, recent works have focused on self-supervised monocular depth estimation with synchronized stereo pairs or monocular video. These methods aim to produce depth maps with stereo matching of stereo pairs and/or estimation of the ego-motion between temporal image pairs. Despite its challenging nature due to the lack of ground truth depth labels, these approaches have produced promising results in recent literature [13]. In this paper, we extend these methods by exploring self-supervised learning approaches to monocular depth estimation that also produce per-pixel measures of uncertainty.

[*]Equal Contribution

2. Related Work

We review models for self-supervised depth estimation that do not require dense depth labels at training time. We also review various methods for assessing model uncertainty in deep learning models.

2.1. Self-supervised Depth Estimation

Without a ground-truth label for depth values, self-supervised methods leverage image reprojection during training. Using the estimated depth map, these models project the given image to its reprojection pair and aim to minimize the image reconstruction error. This image reprojection is usually done with stereo pairs of a given scene or monocular image sequences (i.e. monocular videos).

2.1.1 Stereo Depth Estimation

Stereo pair images consist of two views of a given scene side by side. A self-supervised stereo depth estimation model can be trained on stereo pairs of images to estimate the disparity map between the left and the right image. Then, this disparity map is leveraged to compute the reprojected (synthesized) left image from the right image (or vice versa), and the model is trained with the reprojection error. At inference time, the depth can be computed by scaling the reciprocal of the disparity values. Garg *et al.* [11] proposed a model architecture for continuous disparity estimation, and added a smoothness loss term to improve the continuities of the disparity map. Monodepth [14] extended this work by introducing an additional term for left-right depth consistency, reporting successful results comparable to other supervised methods. Other papers expanded upon this work in various ways, such as adding an additional consistency term with trinocular assumptions [22] and using deep generative models [2, 20, 19] for stereo image reconstruction.

2.1.2 Monocular Depth Estimation

Monocular depth estimation methods seek to estimate the depth map and optionally the ego-motion between temporal image pairs. Similar to self-supervised stereo methods, they are trained by the image reprojection error. Often, these models have both depth networks and pose networks to jointly learn to estimate depth values and relative pose change caused by moving cameras [32].

State-of-the-art monocular methods have reported similar performance to self-supervised stereo methods by constraining estimated depth values with predicted surface normals [30], enforcing edge consistency [29], or introducing a geometry-based loss for temporal depth consistency with point-cloud alignment [17]. Monodepth2 [13] proposed a set of improvements for robust computation for reprojection loss in monocular depth estimation, including a) a minimum reprojection loss that can robustly handle occlusions, b) a multi-scale sampling method for loss computation, and c) a binary per-pixel mask to exclude pixels that violate camera motion assumptions during training.

Recent works also focus on leveraging semantic segmentation to improve monocular depth estimation results. Specifically, Ramirez *et al.* [23] used an additional decoder for semantic segmentation to improve smoothness of the predicted disparity map, where as Chen *et al.* [4] introduced a semantic consistency term with semantic segmentation for region-aware depth estimation.

2.2. Estimating Model Uncertainty

Various methods have been proposed to represent model uncertainty in neural networks. Bayesian neural networks [16, 5] (BNN) are a well known approach to assessing model uncertainty. In BNN, instead of training for fixed weights in each layer, we aim to train weights and biases with a probability distribution. Various works showed that Bayesian neural networks can handle overfitting problems by extending standard networks with posterior inference [15].

One popular approach for representing model uncertainty is the usage of Monte Carlo (MC) dropout layers. Gal *et al.* [10] leveraged dropout layers during both training and inference time (also known as Monte Carlo (MC) dropout layers) as a regularization term to estimate the prediction uncertainty. During training time, dropout randomly drops some units of the neural network model to avoid them from overfitting. During test time, dropout layers randomly drop some units as well, resulting in a set of predictions with which we can estimate an approximate distribution and uncertainty. Several papers extended this work in different directions, particularly in vision-related tasks. Wang *et al.* [27] proposed a test-time augmentation-based uncertainty estimation to analyze the effect of different transformations of the input image on the segmentation output. Nair *et al.* [18] applied the MC dropout layers to compute different kinds of uncertainties.

Another intuitive yet powerful method for uncertainty quantification is using model ensembles. Unlike data augmentation or dropout methods, in an ensemble approach we train separate models and combine them at inference time. Abdar *et al.* [1] note that ensemble models can improve performance while also enabling us to estimate a distribution of predictions.

Finally, there is some prior work focused on estimating uncertainty for depth estimation in particular. [21] applies some of the previously mentioned techniques to depth estimation, including dropout sampling and ensembles.

3. Technical Approach

We have two main goals in this work. First, we will implement a self-supervised training algorithm for a neural network that predicts pixelwise depth maps. Second, we will evaluate various methods to produce measures of uncertainty. These methods will compute a per-pixel depth value (usually expressed as mean of a set of inferences) as well as a per-pixel uncertainty (usually expressed as variance). Then, we will evaluate these methods qualitatively (i.e. do regions of high uncertainty correlate with challenging sections of the input image?) and quantitatively (i.e. standard metrics of regression performance like RMSE, and whether high uncertainty correlates with regression error).

3.1. Datasets

For training, we will rely on the well-known KITTI dataset [12]. This dataset contains depth labels which can be used for quantitative evaluation, though the labels will be ignored during training as we leverage self-supervised depth estimation models. For evaluation, we use the KITTI-Eigen evaluation split, which is a common benchmark introduced in [9].

3.2. Evaluation

For evaluation, we will rely on similar metrics that are used in Casser et al. [3]. Consider an $M \times N$ RGB image. Let D_{ij} represent the ground truth depth value at the pixel indexed at the i-th row and j-th column. Let $\hat{D}_{ij}$ represent the predicted depth value. We computed the following per-image metrics on the evaluation set:

- L_1 **error**:

$$\sum_{i=1}^{M}\sum_{j=1}^{N}|D_{ij}-\hat{D}_{ij}|$$

- **RMSE**:

$$\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(D_{ij}-\hat{D}_{ij})^2}$$

3.3. Self supervised depth estimation

For our self-supervised training procedure, we will adopt the training scheme described in Zhao et al. [31], Section B. We set up the learning problem by training a network to predict the appearance of a target image from the viewpoint of another image.

Instead of using labels to train, we will leverage the geometric constraints between consecutive frames, based on projection. Let $T_{t\to t'}$ denote the relative pose for each source view $I_{t'}$ with respect to a target image I_t. The model will predict a depth map D_t that minimizes the photometric reprojection error L_p, where

$$L_p = \sum_{t'} pe(I_t, I_{t'\to t}),$$

and

$$I_{t\to t'} = I_{t'}\text{<}\text{proj}(D_t, T_{t\to t'}, K)\text{>}.$$

Here, pe is the photometric reconstruction error, proj() are the 2D coordinates of the projected depths D_t in $I_{t'}$ and <> is the sampling operator. Following [3], we use L1 and SSIM to define the photometric reconstruction error:

$$pe(I_a, I_b) = \frac{\alpha}{2}(1-\text{SSIM}(I_a, I_b)) + (1-\alpha)||I_a - I_b||_1.$$

As proposed in [28], the structural similarity index (SSIM) is calculated as follows for two $N \times N$ images x and y:

$$SSIM(x,y) = \frac{(2\mu_x mu_y + c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}.$$

Here, μ_x is the mean of x, μ_y is the mean of y, σ_x^2 is the variance of x, σ_y^2 is the variance of y, σ_{xy} is the covariance of x, y. Also, $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$, L is the dynamic range of the pixel values. The constants used are $k_1 = 0.01$, $k_2 = 0.03$, $\alpha = 0.85$, which was used in [13].

Lastly, to improve the smoothness of the depth map, we leverage the smoothness term proposed in [14]. Since there are infinitely many possible depth values that can correctly reconstruct the novel view, this edge-aware smooth term is important in training due to this ambiguous nature of self-supervised depth estimation. The smoothness error is defined as the following:

$$L_s = |\partial_x d_t^*|\, e^{-|\partial_x I_t|} + |\partial_y d_t^*|\, e^{-|\partial_y I_t|},$$

where $d_t^* = d_t/\overline{d_t}$ is the mean-normalized inverse depth, and ∂_x, ∂_y denote the image gradients.

Finally, the combined training loss can be defined as the weighted sum of the photometric loss and the smoothness loss:

$$L = \mu L_p + \lambda L_s.$$

We define μ as a binary per-pixel mask to only include the loss of pixels in which the reprojection error of the reconstructed image $I_{t'\to t}$ is lower than that of the original, by defining μ as $[\min_{t'} pe(I_t, I_{t'\to t}) < \min_{t'} pe(I_t, I'_t)]$. We used a value of 0.001 for λ, as in [13].

3.4. Model Architecture

For the architecture, we will use the well known fully-convolutional U-Net [24] to predict depth with a ResNet18 backbone pretrained on ImageNet. As done in Godard et al. [13], we will also use a separate pose network to estimate transformation between poses of a pair of frames, which also uses a ResNet18 backbone.

This produces a depth map like the one shown below:

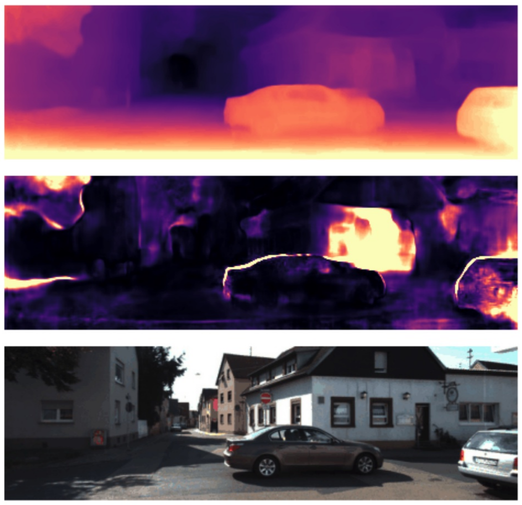

Figure 1. Row 1: Example pixelwise depth map. Row 2: Example pixelwise variance map. Row 3: Input image. This is an example result from ENSEMBLEDEPTH.

3.5. Uncertainty estimation

We will evaluate three different approaches to uncertainty estimation: ENSEMBLEDEPTH, DROPOUTDEPTH, and AUGMENTDEPTH. All of the below methods are trained with PyTorch, for 20 epochs with batch size 11. We use a learning rate of 10^{-4} for the first 15 epochs, and 10^{-5} for the remaining 5 epochs, which was also used in [13]. We also use the same augmentations as [13], namely horizontal flips, random brightness, contrast, saturation, and hue jitter with respective ranges of ±0.2, ±0.2, ±0.2, and ±0.1. All images have resolutions 640×192.

3.5.1 EnsembleDepth

Training time. At training time, we will train $N = 3$ models for 20 epochs on KITTI.
Inference time. At inference time, we will run one inference of each of the N models and compute the mean and variance of the pixel depth prediction.

In Algorithm 1 and all of the below pseudocode listings, GetMean and GetVariance are functions that accept a list of predictions, each of the same dimensions of the input image resolution, and compute the per-pixel mean and variance.

Algorithm 1 EnsembleDepth: Uncertainty via ensemble

```
procedure EnsembleDepth($I, N$)
    Predictions ← []
    for i in [1..M] do
        M ← GetModel(i)
        $D_i$ ← M.Inference(I)
        Predictions.append($D_i$)
    MeanDepth ← GetMean(Predictions)
    VarDepth ← GetVariance(Predictions)
    return MeanDepth, VarDepth
```

3.5.2 DropoutDepth

Training time. At training time, we train a single model with dropout $p = 0.1$ for 20 epochs on KITTI. In this setting, it is important that we train the model with dropout [25]. We added a dropout layer after each convolutional layer of the depth decoder, resulting in 10 dropout layers.
Inference time. At inference time, we will run N inferences of the model with dropout $p = 0.1$, and compute mean and variance statistics.

Algorithm 2 DropoutDepth: Uncertainty via model dropout

```
procedure DropoutDepth($I, N, p$)
    Predictions ← []
    M ← InitDropoutModel($p$)
    for i in [1..N] do
        $D_i$ ← M.Inference(I)
        Predictions.append($D_i$)
    MeanDepth ← GetMean(Predictions)
    VarDepth ← GetVariance(Predictions)
    return MeanDepth, VarDepth
```

3.5.3 AugmentDepth

Training time. At training time, we train a single model for 20 epochs on KITTI.
Inference time. We apply each of the following transforms:

- The identity
- Horizontal flip
- Random color jitter #1 with max brightness = 0.25, max contrast = 0.25, max saturation = 0.25, max hue = 0.25.
- Random color jitter #2 with max brightness = 0.25, max contrast = 0.25, max saturation = 0.25, max hue = 0.25.
- Gaussian blur with 3×3 kernels, standard deviations of 0.1.
- Gaussian blur with 5×5 kernels, standard deviations of 0.1.

We run $N = 6$ inferences of the model and compute mean and variance statistics. After some initial experimentation, we found that it was important to only use augmentations that preserved the realism of the scene. For example vertical flips are unsuitable because the sky ends up on the ground, which is not a realistic scenario represented in the dataset. Similarly, color jittering with an intensity that was too large decreased the quality of the predictions.

Algorithm 3 AugmentDepth: Uncertainty via data augmentation

```
procedure AugmentDepth($I, N$)
    Predictions ← []
    M ← InitModel()
    A ← InitRandomAugmentations()
    for i in [1..N] do
        $I_{\text{aug}}$ ← A.RandomAugment($I$)
        $D_i$ ← M.Inference($I_{\text{aug}}$)
        Predictions.append($D_i$)
    MeanDepth ← GetMean(Predictions)
    VarDepth ← GetVariance(Predictions)
    return MeanDepth, VarDepth
```

4. Results and Evaluation

4.1. Qualitative Analysis

In Figures 2, 3, and 4, we provide side by side outputs of each method on the same frame. Qualitatively, we notice a few patterns. Firstly, the mean depth predictions of all of the models are relatively accurate. This is also corroborated in the metrics section below, where we observe that

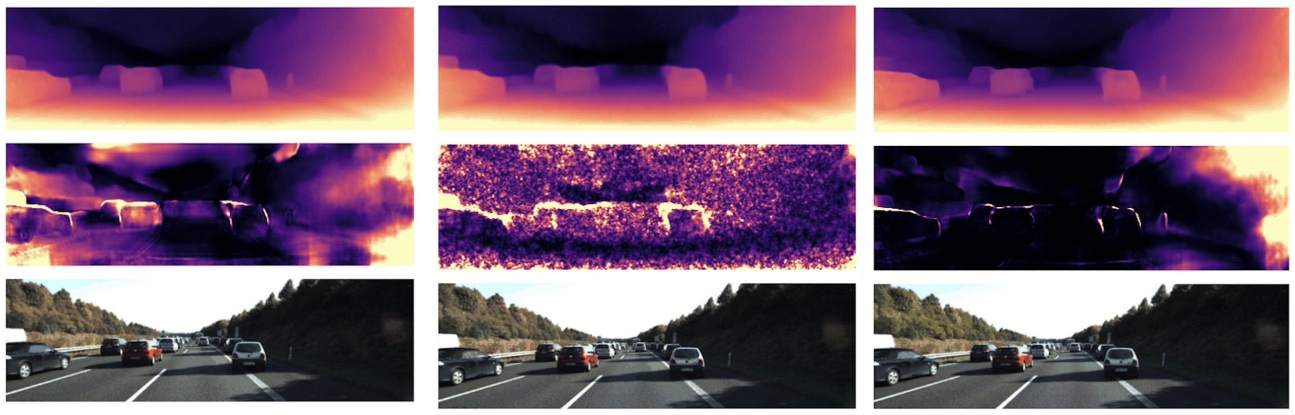

Figure 2. Inferences on 2011_10_03-2011_10_03_drive_0047_sync. From leftmost column to right columns, the algorithms used are ENSEMBLEDEPTH, DROPOUTDEPTH, and AUGMENTDEPTH. Row 1: mean per-pixel depth, Row 2: variance of depth (uncertainty).

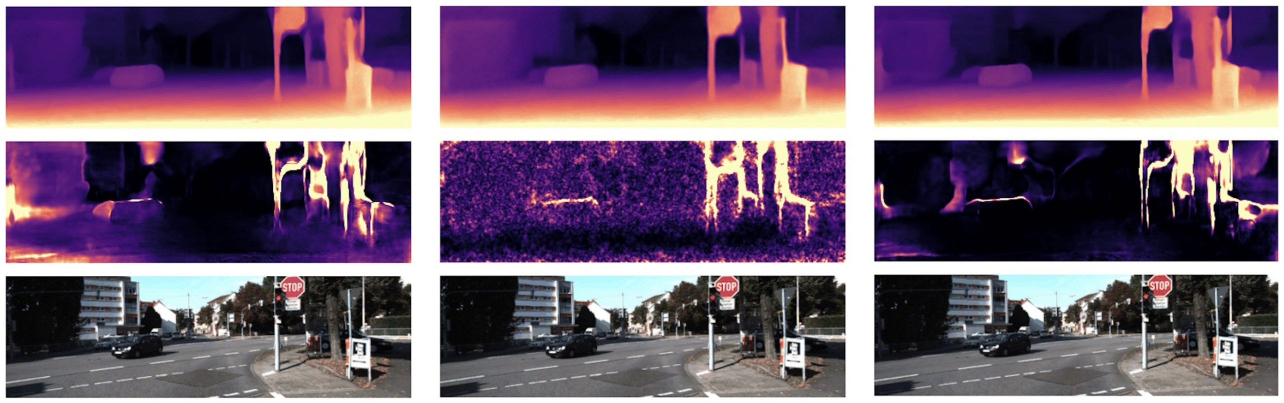

Figure 3. Inferences on 2011_09_26-2011_09_26_drive_0084_sync. From leftmost column to right columns, the algorithms used are ENSEMBLEDEPTH, DROPOUTDEPTH, and AUGMENTDEPTH. Row 1: mean per-pixel depth, Row 2: variance of depth (uncertainty).

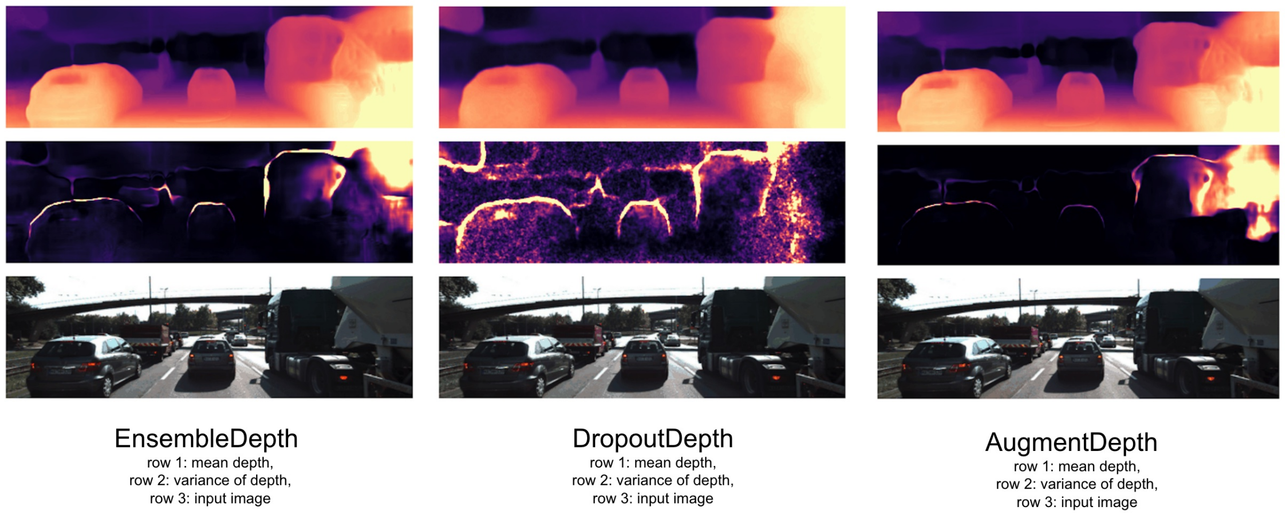

Figure 4. Inferences on 2011_09_26-2011_09_26_drive_0052_sync. From leftmost column to right columns, the algorithms used are ENSEMBLEDEPTH, DROPOUTDEPTH, and AUGMENTDEPTH. Row 1: mean per-pixel depth, Row 2: variance of depth (uncertainty).

each model performs similarly to the Monodepth2 baseline. Looking at the variance, we observe that all models produce higher uncertainty around object boundaries. This is intuitive, since there may be uncertainty about exactly where an object begins and ends in the input image.

We observe that DropoutDepth appears the noisiest, while EnsembleDepth and AugmentDepth output relatively smooth uncertainty maps. This could be attributed to the increased stochasticity of DropoutDepth. EnsembleDepth has a higher variance overall than AugmentDepth. This indicates there is more variability in model outputs across different models trained on the same data than in model outputs from the same model with augmentations applied to the input.

Focusing on the red car in the middle of the frame in Figure 2, we notice that the shape of the uncertainty map surrounding it takes the following shape: ⊓. This makes sense, as opposed to a square shape, because there is greater uncertainty in depth when the boundary shifts to regions other than the ground. For example, moving a slight amount from the top of the car to the sky will result in a very large variance shift, since the sky has effectively infinite depth (due to lack of objects).

Examining the region containing the signs on the right side of Figure 3, we notice that all three methods produce the greatest uncertainty in that area. Looking closer, all three models appear to produce incorrect depth values for the white sign below the stop sign. This is an interesting scenario where high uncertainty indicated by the uncertainty map does correlate with model errors.

Finally, looking at Figure 4, we notice that AugmentDepth and EnsembleDepth produce high uncertainty on the large vehicle on the right. The right side of Figure 3 also has a region with relatively high uncertainty. A possible explanation for this is that it is more difficult for the pose network to predict a transform between consecutive frames when objects go in and out of view (for example, transitioning from being out of the frame to being inside the camera frame).

4.2. Evaluation

We computed mean RMSE and mean variance across 691 images in the KITTI Eigen evaluation set. As seen in Table 1, EnsembleDepth (RMSE 4.666) outperformed the Monodepth2 baseline RMSE, which was 4.863.

4.3. Does model uncertainty correlate with model errors?

An important question we wanted to ask was whether high model uncertainty correlated with RMSE. This has useful applications at inference time, since we may be able to determine scenarios where the model is likely to make an error. We plotted the mean per-pixel variance vs. mean RMSE error for EnsembleDepth, DropoutDepth, and AugmentDepth in Figures 5, 6, and 7. Overall we did find a positive correlation that indicates that high model uncertainty often is correlated with model er-

Algorithm	Mean RMSE	Mean L1	Mean Variance
EnsembleDepth ($N = 3$)	**4.666**	**2.236**	1.916
DropoutDepth ($N = 3$)	4.877	2.332	1.371
DropoutDepth ($N = 5$)	4.852	2.330	1.645
DropoutDepth ($N = 10$)	4.831	2.314	1.845
DropoutDepth ($N = 25$)	4.817	2.304	1.979
DropoutDepth ($N = 50$)	4.812	2.298	2.015
AugmentDepth ($N = 6$)	4.839	2.313	1.122

Table 1. Mean RMSE and mean variance across 691 images in the KITTI Eigen evaluation set.

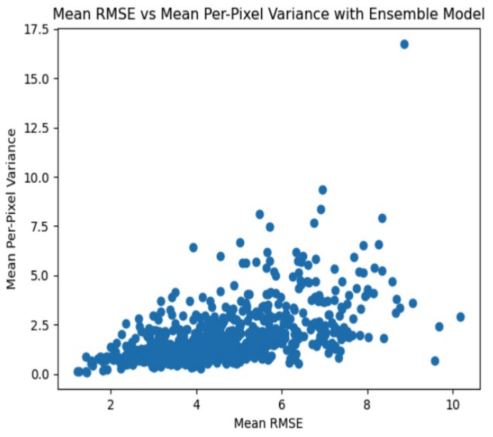

Figure 5. Plot of mean per-pixel variance vs. mean RMSE error for EnsembleDepth. Each point indicates one image from the KITTI (Eigen) evaluation set.

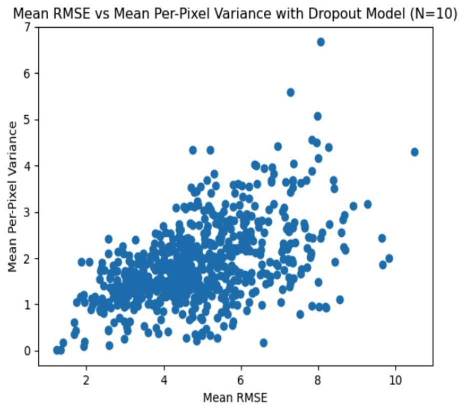

Figure 6. Plot of mean per-pixel variance vs. mean RMSE error for DropoutDepth. Each point indicates one image from the KITTI (Eigen) evaluation set.

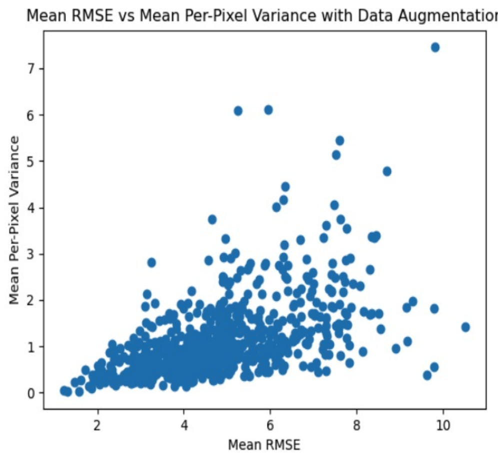

Figure 7. Plot of mean per-pixel variance vs. mean RMSE error for AUGMENTDEPTH. Each point indicates one image from the KITTI (Eigen) evaluation set.

rors – this is visually apparent in the plots. Quantitatively, we computed correlation coefficients of datapoints of mean per-pixel variance vs. mean RMSE error for each model. The mean correlation coefficient was 0.5235 for ENSEMBLEDEPTH, 0.5318 for DROPOUTDEPTH, and 0.5282 for AUGMENTDEPTH.

4.4. Varying the number of inferences in DROPOUTDEPTH

We were interested in evaluating the performance of DROPOUTDEPTH as we varied N, the number of inferences. One investigation we did was sweep N across values in the set $\{3, 5, 10, 25, 50\}$. We investigated the mean RMSE across all images in the KITTI-Eigen evaluation split, as well as the mean per-pixel variance across the same images. These values are shown in Table 1. In Figure 8, we noticed that as N increased, the Mean RMSE decreased monotonically, where the evaluation was performed on the mean depth map produced by DROPOUTDEPTH. This is unsurprising, as increasing N can intuitively be compared to increasing the number of models in an approximate ensemble. We also observed that the mean per-pixel variance increased as the number of inferences increased, but started to level off at a value slightly above 2.

5. Discussion

Overall, we found that each of the proposed methods poses different strengths and weaknesses. We found that ENSEMBLEDEPTH gives the lowest error as measured by RMSE. A drawback of ENSEMBLEDEPTH is that it requires training $N = 3$ models separately, requiring a high training

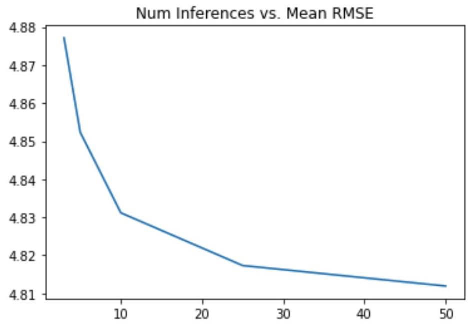

Figure 8. Plot of number of inferences vs. mean RMSE for DROPOUTDEPTH (KITTI Eigen evaluation set). As the number of inferences increases, the quality of the depth map output by the model increases, so error goes down.

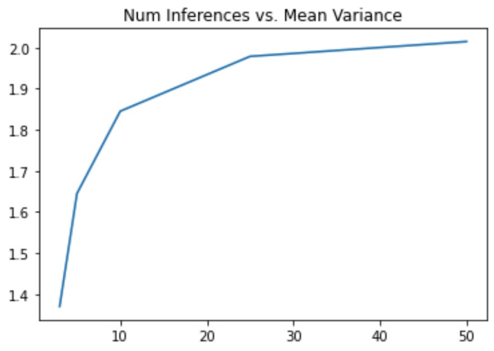

Figure 9. Plot of number of inferences vs. mean per-pixel variance for DROPOUTDEPTH (KITTI Eigen evaluation set). As the number of inferences increases, the mean per-pixel variance also increases.

time compute budget.

We found that DROPOUTDEPTH gives us the strongest correlation between RMSE and model uncertainty (variance), though other models also exhibit a similar, slightly smaller correlation. This correlation allows us to effectively identify potential scenarios where the model may make errors. A drawback of DROPOUTDEPTH is that it requires a large inference-time compute budget, since we found that $N = 5$ inferences (RMSE 4.852) was required to match the performance of the Monodepth2 baseline (RMSE 4.863). However, it is possible to implement DROPOUTDEPTH efficiently by running N inferences in parallel with batch size N, so the actual impact on inference latency depends on the implementation details.

Finally, AUGMENTDEPTH produces qualitatively similar uncertainty results to ENSEMBLEDEPTH, but is a middle ground computationally, requiring 6 inferences. AUGMENTDEPTH had lower RMSE than DROPOUTDEPTH with $N = 3$ and $N = 5$, but had higher RMSE than all the other models.

6. Conclusion and Future Work

In conclusion, all of the methods presented have characteristics and tradeoffs that are useful in different scenarios. The method that minimized RMSE on the evaluation set was ENSEMBLEDEPTH, so this may be the best method to use for model deployment to maximize performance. To identify potential model errors with a low training time and high inference time compute budget, DROPOUTDEPTH is a good choice. A computational middle ground is AUGMENTDEPTH, which requires a low training time budget and a medium inference time budget.

There are various potential future work items. One possibility is exploring different backbone architectures, like vision transformers, which have recently been applied to produce good results on image classification tasks [8]. We could explore different augmentation policies, such as learned augmentation policies (e.g. AutoAugment [6]) or generative models for data augmentations. We could further examine different approaches to implement dropout, such as adding dropout layers at different parts of the model graph, and training DROPOUTDEPTH with different dropout probabilities.

Another possible extension of our experiment is to use Monte Carlo batch normalization [26], which can be used to estimate model uncertainty with batch normalization layers as an approximate Bayesian model. As Do *et al.* [7] suggested, we can apply both MC dropout layers and MC batch normalization layers for computing prediction uncertainty.

Finally, we could explore the use of other descriptive statistics beyond the variance to quantify uncertainty; for example, the number of inferences beyond k standard deviations from the mean depth prediction.

7. Supplemental Materials

Our code is available at `https://github.com/acganesh/monodepth-uncertainty`. We have also made videos of each model available at

- ENSEMBLEDEPTH: `https://www.youtube.com/watch?v=CgzsFZUNcFg`
- DROPOUTDEPTH: `https://www.youtube.com/watch?v=EMYCOdXZnsE`
- AUGMENTDEPTH: `https://www.youtube.com/watch?v=aSxD-TvaeUw`

8. Acknowledgments

We thank the authors of [13] for making the `monodepth2` open source code repository available, which we adapted to fit our needs.

Appendix

In all images to follow, row one is the mean per-pixel depth prediction, row two is the model uncertainty (variance of depth), and row three is the input image.

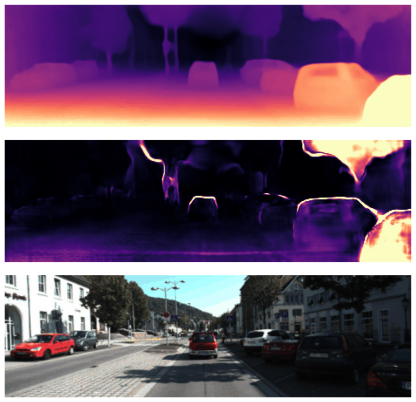
Figure 10. ENSEMBLEDEPTH on KITTI 2011_09_26-2011_09_26_drive_0059_sync.

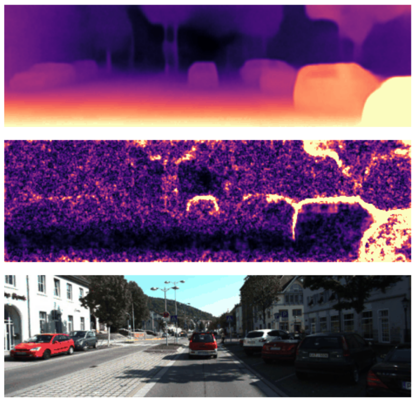
Figure 11. DROPOUTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0059_sync.

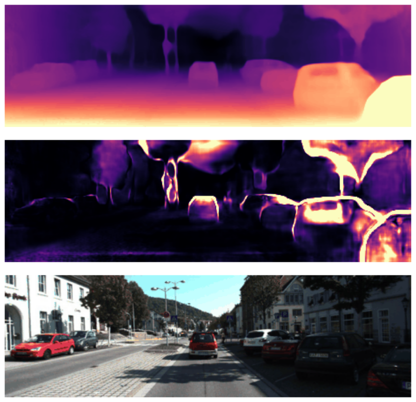
Figure 12. AUGMENTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0059_sync.

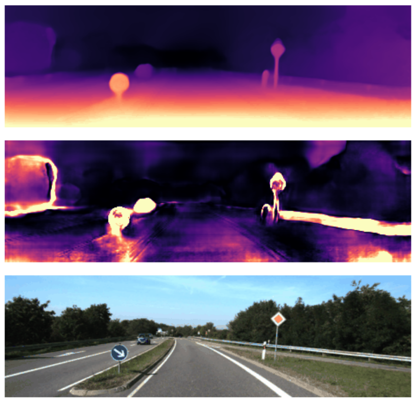
Figure 13. ENSEMBLEDEPTH on KITTI 2011_09_26-2011_09_26_drive_0029_sync.

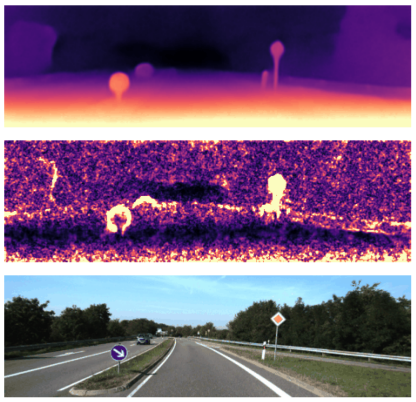
Figure 14. DROPOUTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0029_sync.

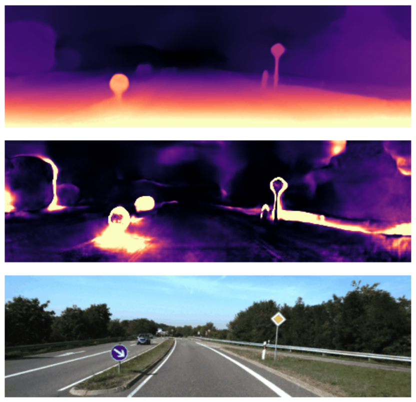
Figure 15. AUGMENTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0029_sync.

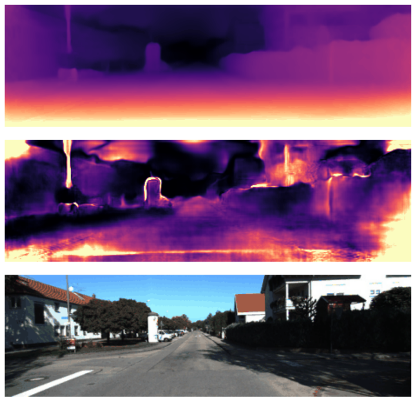
Figure 16. ENSEMBLEDEPTH on KITTI 2011_09_30-2011_09_30_drive_0018_sync.

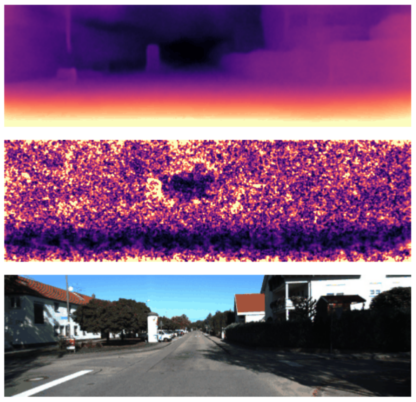
Figure 17. DROPOUTDEPTH on KITTI 2011_09_30-2011_09_30_drive_0018_sync.

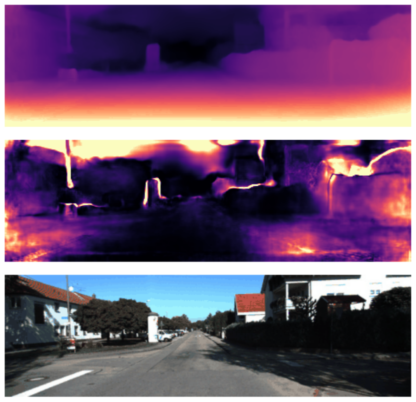
Figure 18. AUGMENTDEPTH on KITTI 2011_09_30-2011_09_30_drive_0018_sync.

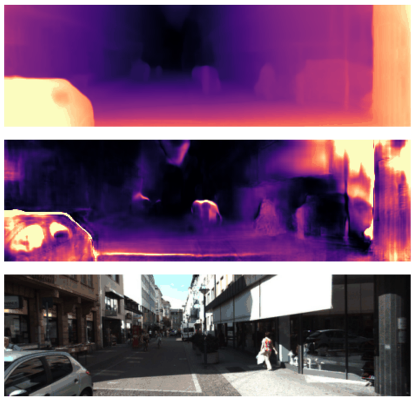
Figure 19. ENSEMBLEDEPTH on KITTI 2011_09_29-2011_09_29_drive_0071_sync.

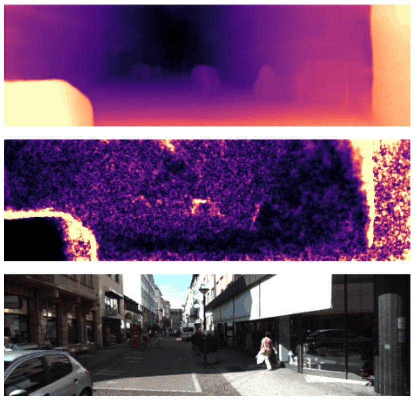
Figure 20. DROPOUTDEPTH on KITTI 2011_09_29-2011_09_29_drive_0071_sync.

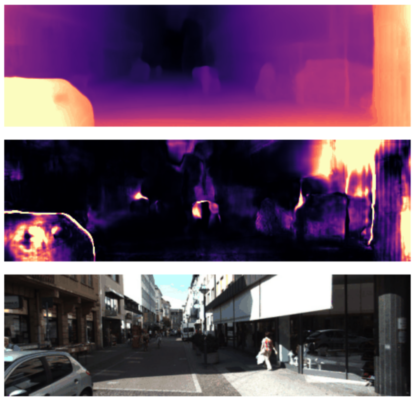
Figure 21. AUGMENTDEPTH on KITTI 2011_09_29-2011_09_29_drive_0071_sync.

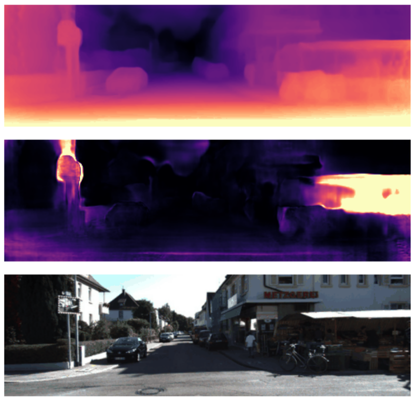
Figure 22. ENSEMBLEDEPTH on KITTI 2011_09_30-2011_09_30_drive_0027_sync.

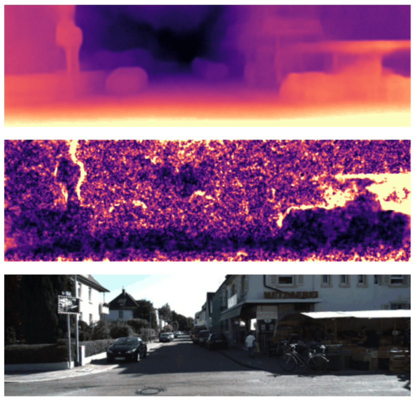
Figure 23. DROPOUTDEPTH on KITTI 2011_09_30-2011_09_30_drive_0027_sync.

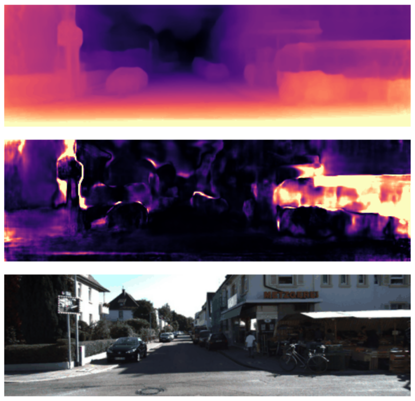
Figure 24. AUGMENTDEPTH on KITTI 2011_09_30-2011_09_30_drive_0027_sync.

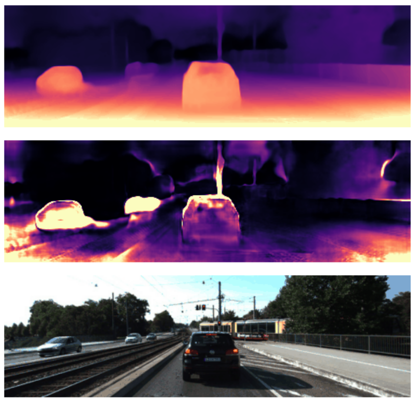
Figure 25. ENSEMBLEDEPTH on KITTI 2011_09_26-2011_09_26_drive_0101_sync.

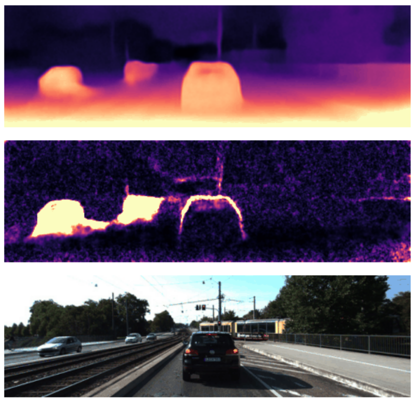
Figure 26. DROPOUTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0101_sync.

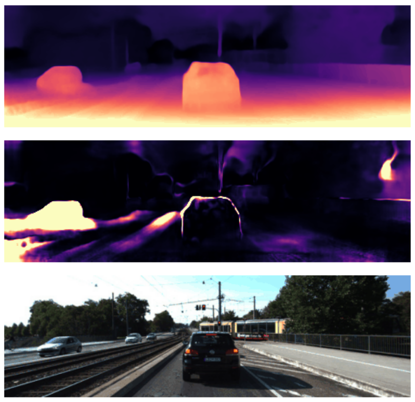
Figure 27. AUGMENTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0101_sync.

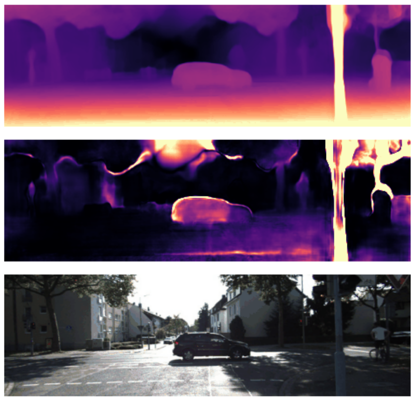
Figure 28. ENSEMBLEDEPTH on KITTI 2011_09_26-2011_09_26_drive_0096_sync.

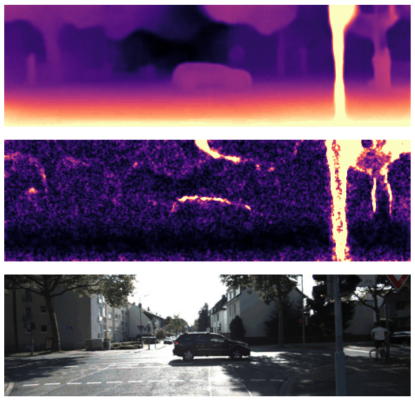
Figure 29. DROPOUTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0096_sync.

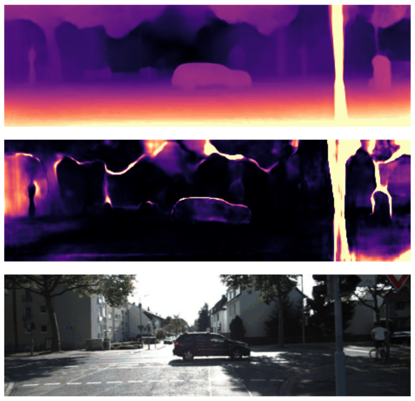
Figure 30. AUGMENTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0096_sync.

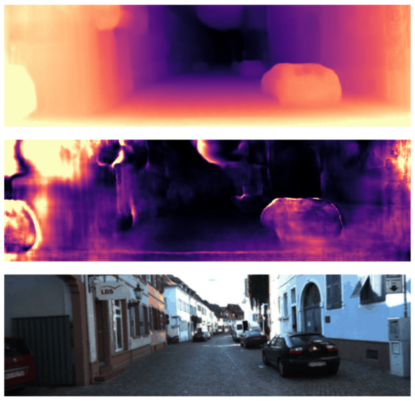
Figure 31. ENSEMBLEDEPTH on KITTI 2011_09_26-2011_09_26_drive_0093_sync.

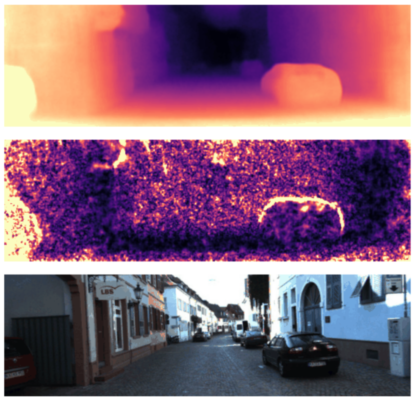
Figure 32. DROPOUTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0093_sync.

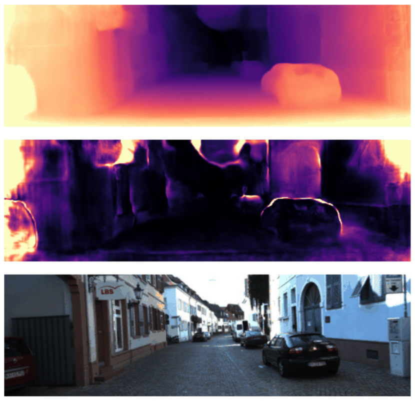
Figure 33. AUGMENTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0093_sync.

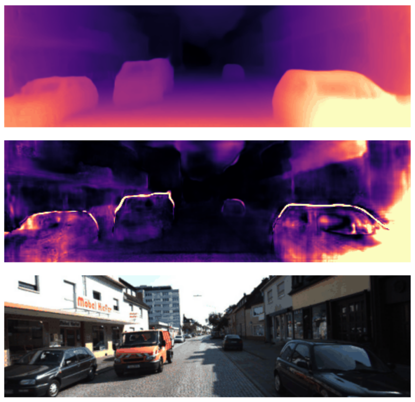
Figure 34. ENSEMBLEDEPTH on KITTI 2011_09_26-2011_09_26_drive_0048_sync.

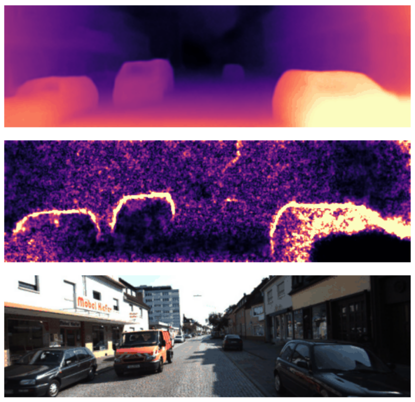
Figure 35. DROPOUTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0048_sync.

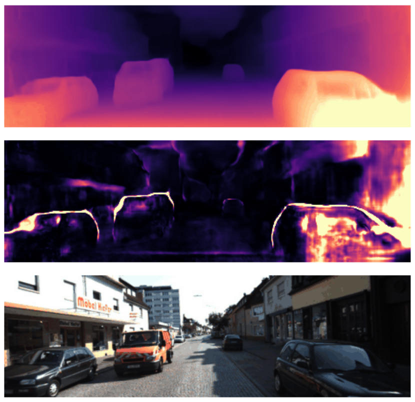
Figure 36. AUGMENTDEPTH on KITTI 2011_09_26-2011_09_26_drive_0048_sync.

References

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. W. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *CoRR*, abs/2011.06225, 2020.

[2] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia. Generative adversarial networks for unsupervised monocular depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[3] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *CoRR*, abs/1811.06152, 2018.

[4] P.-Y. Chen, A. H. Liu, Y.-C. Liu, and Y.-C. F. Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[5] T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691, Bejing, China, 22–24 Jun 2014. PMLR.

[6] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[7] H. P. Do, Y. Guo, A. J. Yoon, and K. S. Nayak. Accuracy, uncertainty, and adaptability of automatic myocardial asl segmentation using deep cnn. *Magnetic Resonance in Medicine*, 83:1863 – 1874, 2019.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[10] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2016.

[11] R. Garg, V. K. B.G., G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 740–756, Cham, 2016. Springer International Publishing.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[13] C. Godard, O. M. Aodha, and G. J. Brostow. Digging into self-supervised monocular depth estimation. *CoRR*, abs/1806.01260, 2018.

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

[15] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *J. Mach. Learn. Res.*, 18:14:1–14:45, 2017.

[16] D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 05 1992.

[17] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 5667–5675. Computer Vision Foundation / IEEE Computer Society, 2018.

[18] T. Nair, D. Precup, D. Arnold, and T. Arbel. Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. *Medical Image Analysis*, 59:101557, 09 2019.

[19] A. Pilzer, S. Lathuilière, D. Xu, M. M. Puscas, E. Ricci, and N. Sebe. Progressive fusion for unsupervised binocular depth estimation using cycled networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[20] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe. Unsupervised adversarial depth estimation using cycled generative networks. In *2018 International Conference on 3D Vision (3DV)*, pages 587–595. IEEE, 2018.

[21] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia. On the uncertainty of self-supervised monocular depth estimation, 2020.

[22] M. Poggi, F. Tosi, and S. Mattoccia. Learning monocular depth estimation with unsupervised trinocular assumptions. In *2018 International Conference on 3D Vision (3DV)*, pages 324–333, Los Alamitos, CA, USA, sep 2018. IEEE Computer Society.

[23] P. Z. Ramirez, M. Poggi, F. Tosi, S. Mattoccia, and L. di Stefano. Geometry meets semantics for semi-supervised monocular depth estimation. In *ACCV*, 2018.

[24] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[26] M. Teye, H. Azizpour, and K. Smith. Bayesian uncertainty estimation for batch normalized deep networks. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4907–4916. PMLR, 10–15 Jul 2018.

[27] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 02 2019.

[28] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similar-

ity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[29] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia. LEGO: learning edge with geometry all at once by watching videos. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 225–234. Computer Vision Foundation / IEEE Computer Society, 2018.

[30] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised learning of geometry from videos with edge-aware depth-normal consistency. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 7493–7500. AAAI Press, 2018.

[31] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian. Monocular depth estimation based on deep learning: An overview. *CoRR*, abs/2003.06620, 2020.

[32] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. *CoRR*, abs/1704.07813, 2017.