Bayesian Deep Basis Fitting for Depth Completion with Uncertainty

Chao Qu   Wenxin Liu   Camillo J. Taylor

University of Pennsylvania

{quchao, wenxinl, cjtaylor}@seas.upenn.edu

Abstract

In this work we investigate the problem of uncertainty estimation for image-guided depth completion. We extend Deep Basis Fitting (DBF) [54] for depth completion within a Bayesian evidence framework to provide calibrated per-pixel variance. The DBF approach frames the depth completion problem in terms of a network that produces a set of low-dimensional depth bases and a differentiable least squares fitting module that computes the basis weights using the sparse depths. By adopting a Bayesian treatment, our Bayesian Deep Basis Fitting (BDBF) approach is able to 1) predict high-quality uncertainty estimates and 2) enable depth completion with few or no sparse measurements. We conduct controlled experiments to compare BDBF against commonly used techniques for uncertainty estimation under various scenarios. Results show that our method produces better uncertainty estimates with accurate depth prediction.

Figure 1: Qualitative results of our method, Bayesian Deep Basis Fitting (BDBF), which outputs uncertainty estimates with depth completion.

1. Introduction

As we seek to incorporate learned modules in safety critical applications such as autonomous driving, reliable uncertainty estimation becomes as critical as prediction accuracy [59]. Depth completion is one such task where well-calibrated uncertainty estimates can help to enable robust machine perception. Deep Convolutional Neural Networks (CNNs) are commonly used to solve structured regression problems like depth prediction due to their strong expressive power and inductive bias [12]. However, in its native form, a CNN only produces a point estimate, which offers few insights into whether or where its output should be trusted. Many probabilistic deep learning methods have been proposed to address this issue [44, 17], but they often fail to output calibrated uncertainty [23] or become susceptible to distributional shift [50]. Moreover, these methods can be expensive to compute due to the need for test time sampling [18] or inference over multiple models [36].

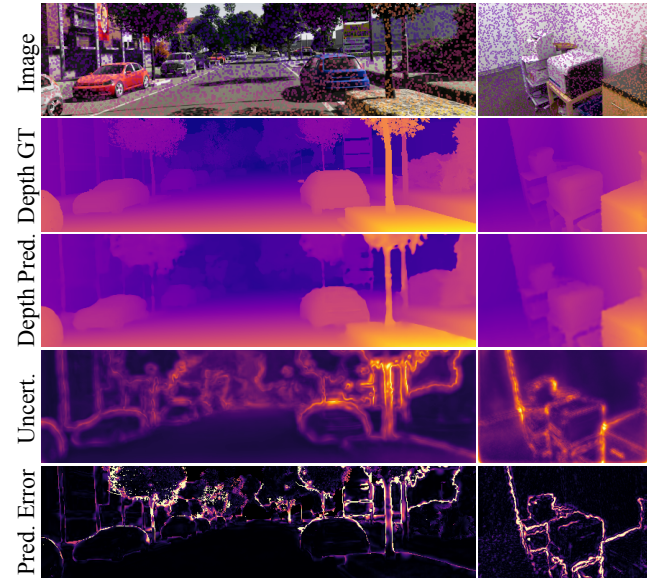In this work, we propose a method for depth completion with uncertainty estimation that avoids the above limitations. Our approach builds on the idea of Deep Basis Fitting (DBF) [54]. DBF replaces the last layer of a depth completion network with a set of *data-dependent* weights. These weights are computed by a differentiable least squares fitting module between the penultimate features and the sparse depths. The network can also be seen as an adaptive basis function which explicitly models scene structure on a low-dimensional manifold [4, 60]. It can be used as a replacement to the final layer (with no change to the rest of the network or training scheme), which greatly improves depth completion performance.

We extend DBF by formulating it within a Bayesian evidence framework [3]. This is done by placing a prior distribution on the DBF weights and marginalizing it out during inference. Such last-layer probabilistic approach have been shown to be reasonable approximations to full Bayesian Neural Networks [34], while providing the advantage of tractable inference [48]. This is conceptually similar to

Neural Linear Models (NLMs) [58] with the notable distinction that we perform Bayesian linear regression on each image as opposed to the entire dataset.

A Bayesian treatment also enables depth completion with highly sparse data. In DBF, when the number of sparse depths falls below the dimension of the bases, the underlying linear system becomes under-determined. We show that by learning a shared prior across images, our method is able to handle any number of sparse depth measurements.

We name our approach Bayesian Deep Basis Fitting (BDBF) and summarize its advantages: 1) It can be used as a drop-in replacement to the final layer of many depth completion networks and outputs uncertainty estimates (in the form of per-pixel variance). 2) Compared to other uncertainty estimation techniques, it produces higher quality uncertainty with one training session, one saved model and one forward pass, without needing extra parameters or modifications to the loss function. 3) It can handle any sparsity level, with performance degrading gracefully towards a pure monocular method when the number of depth measurements goes to zero.

2. Related Work

2.1. Uncertainty Estimation for Neural Networks

We start by reviewing uncertainty estimation techniques for neural networks. There are two types of uncertainty that one could model: *epistemic (model)*, which describe the uncertainty in the model and *aleatoric (data)*, which reflects the inherent noise in the data [33]. Modeling uncertainty in neural networks can be achieved by placing probabilistic distributions on network weights. Such networks are called Bayesian Neural Networks (BNN) [44]. Direct inference in BNNs is intractable for continuous variables, and different approximation techniques have been explored [44, 29, 22, 5, 46, 45, 30]. However, they don't scale well to large datasets and complex models, and are thus impractical for current vision tasks.

Gal *et al.* [18] proposed the use of *dropout* as an approximate variational inference method for BNNs. However, their method requires multiple forward passes to obtain Monte Carlo model estimates at test time. Another research direction is *assumed density filtering* (ADF) [49] which can be viewed as a single Expectation Propagation pass. Gast *et al.* [20] chose to propagate activation uncertainties without probabilistic weights in a lightweight manner, which requires modifying the layer operations based on moment matching.

Predictive methods directly output mean and variance of some parametric distribution by minimizing the negative-log likelihood (NLL) loss [47]. They only require small changes to the original network by adding a variance prediction head. This simplicity makes it a popular choice among

Method	#T	#M	#F	Alea.	Epis.
Predictive [47, 31]	1	1	1	✓	
Dropout (Predictive) [18, 31]	1	1	K	(✓)	✓
Snapshot (Predictive) [26]	1	K	K	(✓)	✓
Bootstrap (Predictive) [36]	K	K	K	(✓)	✓
Proposed (BDBF)	1	1	1	✓	✓

Table 1: Comparison of different uncertainty estimation techniques. The first three columns represent number of training sessions (T), copies of model saved (M), and number of forward passes (F) at test time respectively. Last two columns indicate whether a method estimates data or model uncertainties. Dropout, snapshot and bootstrap ensemble can all be combined with a predictive approach to model data uncertainty. BDBF has the same complexity as predictive methods.

recent works [31, 39].

Ensemble methods either train multiple models independently with different initializations (*bootstrap*) [36] or save several copies of weights at different stages during training (*snapshot*) [26]. These methods only model epistemic uncertainty but can be combined with a predictive one to model data uncertainty. They achieve good performances in various experimental settings [24, 52, 28], but still need multiple inference passes at test time, which makes them less suitable for resource constrained platforms.

Table 1 summarizes the aforementioned approaches and highlights the difference compared to ours. In Sec. 4.1 we describe in detail the methods that we evaluate against.

2.2. Uncertainty Estimation in Depth Completion

Great progress has been made in the past few years on depth completion ranging from high-density completion for RGB-D/ToF cameras [38, 15, 68, 70], to mid-density completion from LiDAR sensors [64, 43, 10, 61, 9, 8, 71, 37, 67, 69]. Recently, there has also been rising interests in low-density completion from map points generated by Visual-SLAM or Visual-Inertial Odometry [65, 66, 55, 72]. Unlike systems that are designed for a particular sparsity or sensing modality, our proposed method can be seen as a general component for depth completion similar to DBF [54].

A complete review of depth completion literature is out of the scope of this work, we instead focus on methods that also **estimate uncertainty**. Gansbeke *et al.* [19] predict depth and confidence weights for both color and depth branch and fuse them based on the confidence maps. Qiu *et al.* [53] adopt a similar strategy, but additionally guide the depth branch via surface normal prediction. Xu *et al.* [67] use a shared encoder and multiple decoders to predict surface normal, coarse depth and confidence, then use an anisotropic diffusion process to produce refined depth. Park *et al.* [51] instead use a single encoder-decoder network to predict initial depth, affinities and confidence be-
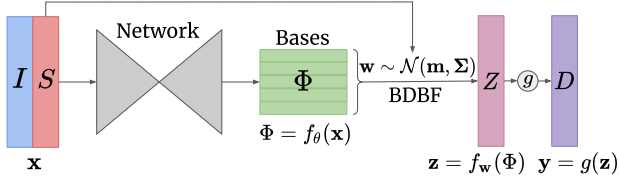
Figure 2: An overview of BDBF for depth completion. The input to the network is an RGB image I and optionally a sparse depth map S. The network produces a M-dimensional depth bases $\boldsymbol{\Phi}$, which has the same spatial resolution as I. BDBF then solves for the weights $\mathbf{w}$ given the sparse depth at valid pixel locations. $\mathbf{w}$ can then be used to reduce the bases into a single channel latent prediction Z, before going through the activation g to produce depth D.

fore applying a non-local spatial propagation to produce the final depth. Note that the uncertainties produced by the above methods are not calibrated and are only used internally. Therefore, they are not readily useful to downstream tasks that require probabilistic reasoning. This type of uncertainty estimation can also be seen as a simplified version of the *predictive* method in [47] without the NLL loss.

Few works have tried to evaluate the quality of depth completion uncertainty. Eldesokey *et al*. [13] present a probabilistic normalized convolution [14] that estimates confidence of both input sparse depths and output dense prediction, for unguided depth completion. Gustafsson *et al*. [24] compared several uncertainty estimation methods applied to depth completion in the same spirit as [52]. We follow their approach and provide a systematic comparison of our proposed method against the best performing schemes from [24, 52] and demonstrate superior performance and efficiency across a range of datasets.

3. Method

3.1. Problem Formulation

Let $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N_{\mathcal{D}}}$ be a dataset containing $N_{\mathcal{D}}$ samples. We wish to learn a neural network f that maps $\mathbf{x}$ to $\mathbf{y}$. In depth completion, the input $\mathbf{x}$ is usually an image and sparse depth pair (I, S), and the output $\mathbf{y}$ is the predicted depth map D. We refer to f_θ as the basis network and its output $\boldsymbol{\Phi}$ a set of *depth bases* [54]. $\boldsymbol{\Phi}$ is then reduced by a *linear* layer $f_\mathbf{w}$ to $\mathbf{z}$, which is then mapped to positive depth values via a *nonlinear* activation function g.

$$\mathbf{y} = f(\mathbf{x}) = g \circ f_\mathbf{w} \circ f_\theta(\mathbf{x}) = g \circ f_\mathbf{w}(\boldsymbol{\Phi}) = g(\mathbf{z}) \quad (1)$$

With a slight abuse of notation, we call $\mathbf{z}$ the *latent* variable and choose g to be the exponential function [12], so $\mathbf{z}$ effectively corresponds to log depth. An overview of our method is shown in Figure 2.

3.2. Bayesian Deep Basis Fitting

We choose to model the distribution of each pixel in the latent space $\mathbf{z}$ rather than in the target space $\mathbf{y}$, since depth is strictly positive and may span several orders of magnitude [57]. Assuming Gaussian noise in the latent space, we define our model to be

$$z_i = \mathbf{w}^\top \boldsymbol{\phi}_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \beta^{-1}) \quad (2)$$

where $\boldsymbol{\phi}_i$ denotes the basis entries corresponding to the latent pixel value z_i and $\beta \in \mathbb{R}$ is a precision parameter that corresponds to the inverse variance of the noise. Assuming that the errors at each pixel are independent, the likelihood function is

$$p(\mathbf{z}|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I}) \quad (3)$$

here $\boldsymbol{\Phi}$ is the $N \times M$ *design matrix* with N the number of sparse depths and M the dimension of $\mathbf{w}$. It is assembled by extracting the basis entries at the pixel locations specified by S.

Given a suitable prior on the last-layer weights $p(\mathbf{w}) = \mathcal{N}(\mathbf{m}_0, \alpha^{-1}\boldsymbol{\Sigma}_0)$, where $\alpha \in \mathbb{R}$ is a precision parameter to scale the covariance $\boldsymbol{\Sigma}_0$, the posterior distribution of $\mathbf{w}$ can be computed analytically following Bayes' rule [3]:

$$p(\mathbf{w}|\mathbf{x}, \mathbf{z}) = \mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma}) \quad (4)$$
$$\propto \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \alpha^{-1}\boldsymbol{\Sigma}_0) \cdot \mathcal{N}(\mathbf{z}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I}) \quad (5)$$

where the mean and covariance are given by

$$\mathbf{m} = \boldsymbol{\Sigma}(\alpha\boldsymbol{\Sigma}_0^{-1}\mathbf{m}_0 + \beta\boldsymbol{\Phi}^\top\mathbf{z}) \quad (6)$$
$$\boldsymbol{\Sigma} = (\alpha\boldsymbol{\Sigma}_0^{-1} + \beta\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1} \quad (7)$$

The latent predictive distribution for a pixel at test time is

$$p(z_*|\mathbf{x}, \mathbf{z}) = \int p(z_*|\mathbf{w})p(\mathbf{w}|\mathbf{x}, \mathbf{z})d\mathbf{w} \quad (8)$$
$$= \mathcal{N}(z_*|\mathbf{m}^\top\boldsymbol{\phi}_*, \boldsymbol{\phi}_*^\top\boldsymbol{\Sigma}\boldsymbol{\phi}_*) \quad (9)$$

The Gaussian assumption is made solely for the purpose of tractable inference. In practice, the shape of the predictive distribution depends heavily on the loss function. Since we use L1 loss for training, we employ a Laplace distribution as its parametric form for evaluating uncertainty [28, 4]. Additionally, a robust norm like Huber [27] can be applied if outliers are present in the target [63] .

3.3. Training

Loss Function. The standard way of learning a Bayesian regressor is by maximizing the marginal likelihood function with respect to the parameters θ of the basis function f_θ,

$$\log p(\mathbf{z}|\mathbf{x}, \alpha, \beta) = \frac{1}{2}(N\ln\beta + M\ln\alpha - N\ln 2\pi$$
$$- E(\mathbf{m}) + \ln|\boldsymbol{\Sigma}| - \ln|\boldsymbol{\Sigma}_0|) \quad (10)$$
$$E(\mathbf{w}) = \beta\|\mathbf{z} - \boldsymbol{\Phi}\mathbf{w}\|^2 + \alpha\|\mathbf{w} - \mathbf{m}_0\|_{\boldsymbol{\Sigma}_0}^2 \quad (11)$$

where $\|\mathbf{v}\|_{\mathbf{A}} = \mathbf{v}^\top \mathbf{A}^{-1} \mathbf{v}$ is the Mahalanobis norm. This is known in literature as *type 2 maximum likelihood* [3].

Directly maximizing (10) has two practical issues. First, one needs to estimate the hyperparameters α and β, which adds large overhead during training. Second, gradients need to be back-propagated through costly operations like matrix inversion and determinant. Together, they pose challenges to the training phase and often produce empirically similar results to a point estimate [58]. We avoid these issues by assuming sufficient sparse points in training ($N \gg M$). This renders the linear system over-determined, which allows us to treat the prior $p(\mathbf{w})$ as infinitely broad. The solution in (6) therefore reduces to a maximum likelihood (ML) one which can be computed efficiently in one pass [3].

$$\mathbf{w}_{\mathrm{ML}} = (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{z}, \ \beta_{\mathrm{ML}}^{-1} = \frac{1}{N}\|\mathbf{z} - \mathbf{\Phi}\mathbf{w}_{\mathrm{ML}}\|^2 \quad (12)$$

The predicted mean and variance of z_* at training time are given by the following according to (9)

$$\mu = \mathbf{w}_{\mathrm{ML}}^\top \boldsymbol{\phi}, \quad \sigma^2 = \beta_{\mathrm{ML}}^{-1} \boldsymbol{\phi}^\top (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \boldsymbol{\phi} \quad (13)$$

Given the above results, we can minimize the Negative Log-Likelihood Loss (NLL) assuming a Laplace distribution [28], which is defined for a single pixel as

$$-\log p(z|\mu, b) \propto \frac{|\mu - z|}{b} + \log b, \quad 2b^2 = \sigma^2 \quad (14)$$

However, this would still involve back-propagation through a costly matrix inversion as in (12). We find in our experiments that this sometimes causes numerical instability during training and incurs a visible decrease in prediction accuracy. Therefore, we opt to minimize directly the L1 loss for supervised learning. This allows our network to be trained in its original form without suffering from the performance drop caused by the NLL loss [40].

Uncertainty Calibration. Not using the likelihood loss comes with the risk of over-confidence in uncertainty estimation, since there is no explicit penalty on the variance prediction. As the number of parameters in θ is usually on the order of millions, the noise variance β^{-1} will be pushed towards zero [48]. One solution is to regularize θ using an L2 regularization term in the optimizer. This introduces an extra hyperparameter to tune: a small regularization would not prevent overfitting, while a large one will render the feature bases inexpressive [62]. We notice empirically that the amount of overconfidence in our method is consistent during training and validation. Therefore, we take a pragmatic approach and propose to solve this problem in terms of estimator consistency [2], measured by normalized estimation error squared (NEES). For a Laplace distribution, NEES is defined as $\varepsilon = (\mu - z)^2 / b^2$. We record the average NEES $\bar{\varepsilon}$ at training time for the last epoch, and use it to scale the variance accordingly during inference with $\bar{\sigma}^2 = \bar{\varepsilon}\sigma^2$, which

attempts to make the final prediction consistent. Note that the scaling factor is computed entirely at training without any additional data and NEES is not used as a loss function.

Shared Prior. Although the prior $p(\mathbf{w})$ is not used in training, we still need it for inference. Rather than estimating a different prior for each image, we make another simplifying assumption that there exists a shared prior for the entire dataset. This aligns with our observation from experiments that $p(\mathbf{w})$ shows a relatively sharp peak. Given our training strategy, we adopt a frequentist approach and collect all ML solutions of weights $\mathbf{w}_{\mathrm{ML}}$ within one training epoch. The mean, $\mathbf{m}_0$, and covariance, $\mathbf{\Sigma}_0$, can then be computed from this set. Having a shared prior enables robust depth completion from a few sparse depth measurements.

3.4. Inference

Inference follows the standard evidence framework [3]. We use EM [11] to estimate the hyperparameters α and β. The re-estimation equations are obtained by maximizing the expected complete-data log likelihood with respect to α, β

$$\alpha^{-1} = \frac{1}{M}\left(\|\mathbf{m} - \mathbf{m}_0\|_{\mathbf{\Sigma}_0}^2 + \mathrm{tr}(\mathbf{\Sigma}_0^{-1}\mathbf{\Sigma})\right) \quad (15)$$

$$\beta^{-1} = \frac{1}{N}\left(\|\mathbf{z} - \mathbf{\Phi}\mathbf{m}\|^2 + \mathrm{tr}(\mathbf{\Phi}^\top \mathbf{\Phi}\mathbf{\Sigma})\right) \quad (16)$$

where $\mathrm{tr}(\cdot)$ is the matrix trace operator. The re-estimated α and β are then plugged back into (6) and (7) to recompute $\mathbf{m}$ and $\mathbf{\Sigma}$. We initialize this process empirically with $\alpha = 1$ and $\beta = \sqrt{N}$ and set the maximum number of iterations to 8. In practice we reach convergence within 2 to 3 iterations when $N \gg M$, thus incurring only a small computation overhead. In the extreme case when $N \to 0$, we rely on the shared prior alone for a pure monocular prediction.

$$\mu = \mathbf{m}_0^\top \boldsymbol{\phi}, \quad \sigma^2 = \boldsymbol{\phi}^\top \mathbf{\Sigma}_0 \boldsymbol{\phi} \quad (17)$$

4. Evaluation

In this section, we show results on various datasets that our method outperforms baseline approaches in uncertainty estimates with accurate depth prediction, and remains resilient to sparsity change and domain-shift.

4.1. Baselines

We describe three baselines for uncertainty estimation that we compare to, which are shown to have strong performance [52, 24] and can be evaluated under controlled settings. As discussed in Section 3.2, all methods output mean and variance of the latent prediction before $g(\cdot)$, and are trained with either L1 loss or its NLL variant (14).

For empirical methods, we choose Snapshot Ensemble [26] (*snap*) which can be completed in one training session
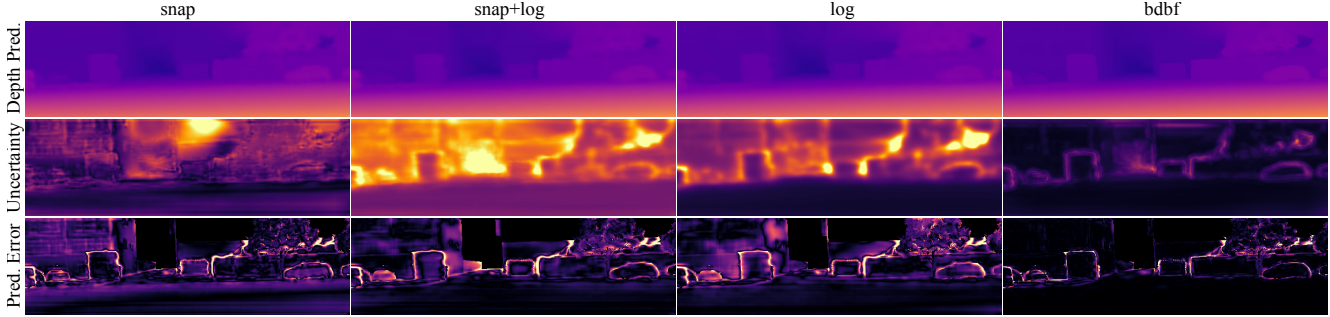
Figure 3: Qualitative results of all methods trained and test with 5% sparsity on VKITTI2.

such that all methods have the same training budget. The mean and variance are computed using K snapshots.

$$\mu = \frac{1}{K}\sum_{i=1}^{K}\mu_i, \quad \sigma^2 = \frac{1}{K}\sum_{i=1}^{K}(\mu_i - \mu)^2 \quad (18)$$

For predictive methods (*log*) [47], we attach a variance prediction head parallel to the depth prediction and train with the NLL loss. Finally, we combine the above two (*snap+log*) [31] to form a predictive ensemble method.

$$\mu = \frac{1}{K}\sum_{i=1}^{K}\mu_i, \quad \sigma^2 = \frac{1}{K}\sum_{i=1}^{K}\left((\mu_i - \mu)^2 + \sigma_i^2\right) \quad (19)$$

4.2. Datasets

Virtual KITTI 2. The VKITTI2 dataset [6] is an updated version to its predecessor [16]. We use sequences 2, 6, 18 and 20 with variations **clone**, **morning**, **overcast** and **sunset** for training and validation, and **clone** in sequence 1 for testing. This results in 6717 training and 447 testing images. The sparse depths are generated by randomly sampling pixels that have a depth less than 80m [24]. Ground truth depths are also capped to 80m following common evaluation protocols. All images are downsampled by half.

NYU Depth V2. The NYU-V2 [56] dataset is comprised of various indoor scenes recorded by an off-the-shelf RGB-D camera. We use the 1449 densely labeled pairs of aligned RGB and depth images. and split it into approximately 75% training and 25% testing. The same depth sampling strategy is adopted as above. Note that we intentionally choose this small dataset (as opposed to the full) to evaluate uncertainty estimation under data scarcity [62].

KITTI Depth Completion. We also evaluate on the KITTI depth completion dataset [64] following its official train/val split. For all experiments other than the official submission, we down sample both the image and depth by half.

4.3. Implementation Details

Network architecture. We use the same basis network for all methods, which is an encoder-decoder architecture with skip connection similar to that in [54]. We use a MobileNet-V2 [57] pretrained on ImageNet [35]. The decoder outputs a set of *multi-scale bases* [54], which are then upsampled to the input resolution and concatenated together to form a final 63-dimensional basis. For baseline methods we initialize the bias of depth prediction head with the average log depth of the dataset and let the variance head predict an initial variance of 1. Our method, however, requires no initialization. When using sparse depths as network input, we adopt the two-stage approach from [66] which first scaffolds the sparse depths by interpolation and then fuses it with the first layer of the encoder via convolution. Note that this depth pre-processing step is orthogonal to the uncertainty estimation techniques and we choose this approach for its simplicity and applicability to both mid- and low-sparsity. All networks use the same setup unless otherwise stated.

Training parameters. For training we use the Adam optimizer [32] with an initial learning rate of 2e-4 and reduce it by half every 5 epochs following [42, 54]. We train our method for 20 epochs and all others for 30. This is to account for the increased training time using our method. For Snapshot Ensemble,we follow the original paper [26] and use the cyclic annealing scheduler from [41] with the same initial learning rate as before. we train for 5 epochs per cycle, and discard the worst snapshot, which leaves us with 5 snapshots. All training is carried out on a single Tesla V100 GPU with the same batch size and random seed. For data augmentation we apply a random horizontal flip with a probability of 0.5 and a small color jitter of 0.02.

4.4. Metrics

Depth prediction metrics. We evaluate depth completion performance using standard metrics [12]. Specifically, we report MAE, RMSE and accuracy (δ-threshold) on depth. Due to space limitation, we only report $\delta_1 < 1.25$.

Uncertainty estimation metrics. Unlike depth prediction which can be compared to ground truth, the true probability density function of depth is not available. This makes evaluating uncertainty estimates a difficult task in itself. Here,

we describe three popular metrics commonly found in literature for uncertainty estimates. Note that each metric has its own advantages and drawbacks, we seek to provide a more comprehensive evaluation by reporting all three.

1) Area Under the Sparsification Error curve (AUSE)↓.
Sparsification plots [1] are commonly used for measuring the quality of uncertainty estimates. Given an error metric (*e.g.* MAE), we sort the prediction errors by their uncertainty in descending order and compute the error metric repeatedly by removing a fraction (*e.g.* 1%) of the most uncertain subset. An oracle sparsification curve is obtained by sorting using the true prediction errors. AUSE is the area between the sparsification curve and the oracle curve. This normalizes the oracle out and can be used to compare different methods [28]. Note that AUSE is a *relative* measure of uncertainty quality, since its computation relies on the *order* of predicted uncertainties.

2) Area Under the Calibration Error curve (AUCE)↓.
For an *absolute* measure of uncertainty estimation quality, [24] proposes to generalize the *Expected Calibration Error (ECE)* [23] metric to regression. For Laplace distributions, given mean μ and variance σ^2, we construct prediction intervals $\mu \pm \Psi^{-1}(\frac{p+1}{2})b$ for $p \in (0,1)$, where Ψ is the CDF of the unit Laplace distribution. For each value of p, we compute the proportion of pixels $\hat{p}$ for which the true target falls within the predicted interval. For a well-calibrated model, $\hat{p}$ should closely match p. The *Calibration Error curve* is defined as $|p - \hat{p}|$, and AUCE is the area under this curve. Like ECE, AUCE is **not** a *proper* scoring rule [50], as there exists trivial solutions which yield perfect scores.

3) Negative Log-likelihood (NLL)↓. NLL (14) is commonly used to evaluate the quality of model uncertainty on a held-out dataset [50]. It is a *proper* scoring rule [21], but over-emphasizes tail probabilities [7] and cannot fully capture posterior in-between uncertainty [62].

4.5. Results

Mid-density Depth Completion. In this setting, we train all methods with 5% sparsity. Table 2 shows quantitative results when testing on the same dataset under the same sparsity level. This is considered an *in-distribution* test. We see significant improvements of our method in almost all metrics. Figure 3 shows qualitative results of one sample from the VKITTI2 test set. Compared to others, *bdbf* not only predicts higher quality depths but also sharper uncertainties that closely match the true prediction errors. This indicates that the learned depth bases in ours is expressive for predicting both depth and uncertainty. Note that when there's enough data, *bdbf* will reduce to *dbf* as expected, but the Bayesian formulation allows for smooth transition to a low sparsity level by incorporating a learned prior.
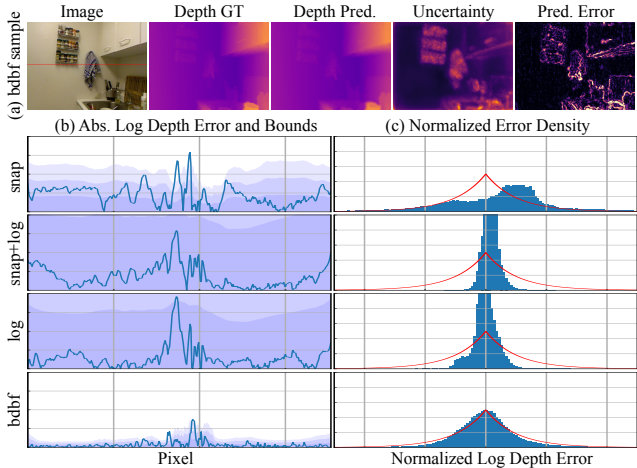

Figure 4: (a) Qualitative results of *bdbf* on one test image from NYU-V2. (b) Absolute log depth error (blue line) and $3b$ bounds (blue shades) for a single row of pixels (red line) from the image. (c) Normalized error density of the entire image compared to a unit Laplace distribution (red line). All axes are of the same scale within each column.
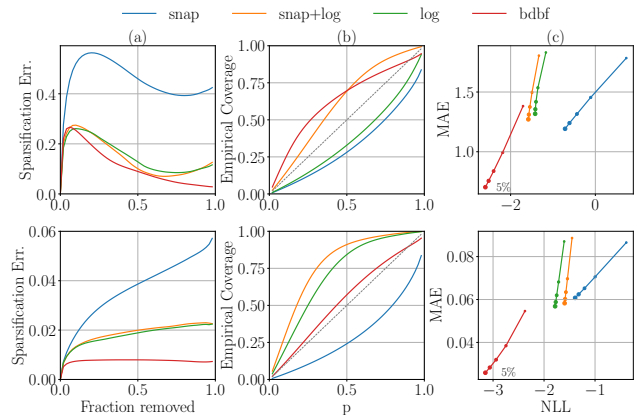

Figure 5: (a) Sparsification error, (b) calibration error and (c) sparsity change plots of NLL *vs.* MAE with 5% sparsity on VKITTI2 (top) and NYU-V2 (bottom). Sparsification and calibration plots are generated using 5% test sparsity. Sparsity change plots are generated with varying test sparsity from 5% to 1%.

We take a closer look at one sample from the NYU-V2 test set in Figure 4 (a), by plotting the absolute prediction error in log space $e = |\mu - z|$ and uncertainty bound $b = \sigma/\sqrt{2}$ for one row of pixels in Figure 4 (b), and the normalized error density $\tilde{e} = e/b$ on the entire image in Figure 4 (c). *bdbf* is the only method where the bound traces the general shape of the prediction error and whose normalized error density resembles that of a unit Laplace distribution. *snap* fails to capture the underlying error dis-

Trained with 5%			VKITTI2						NYU-V2					
Input	Method	%	MAE	RMSE	δ_1	AUSE	AUCE	NLL	MAE	RMSE	δ_1	AUSE	AUCE	NLL
rgbd	snap [26]	5%	1.192	3.267	95.59	0.445	0.170	-0.714	0.061	0.126	99.35	0.036	0.202	-1.390
rgbd	snap+log [31]	5%	1.271	3.432	95.33	0.142	**0.117**	-1.582	0.058	0.123	99.32	0.018	0.256	-1.596
rgbd	log [47]	5%	1.318	3.423	95.37	0.149	0.125	-1.421	0.057	0.121	99.34	0.018	0.210	-1.783
rgbd	dbf [54]	5%	0.709	2.928	97.88	0.148	0.163	-2.488	0.026	0.082	99.64	0.007	0.054	-3.145
rgbd	bdbf	5%	**0.703**	**2.925**	**97.88**	**0.110**	0.136	**-2.596**	**0.026**	**0.082**	**99.64**	**0.007**	**0.039**	**-3.151**

Table 2: Quantitative results of all methods trained and tested with 5% sparsity on VKITTI2 and NYU-V2.

Trained with 500			VKITTI2						NYU-V2					
Input	Method	#	MAE	RMSE	δ_1	AUSE	AUCE	NLL	MAE	RMSE	δ_1	AUSE	AUCE	NLL
rgbd	snap	500	2.312	5.403	90.14	0.459	0.229	-0.207	0.096	0.206	97.53	0.053	0.261	0.211
rgbd	snap+log	500	2.396	5.571	89.88	**0.273**	0.036	-1.150	0.095	0.213	97.44	0.025	0.205	-1.393
rgbd	log	500	2.492	5.800	89.13	0.299	0.095	-0.906	0.097	0.212	97.44	0.025	0.152	-1.502
rgbd	dbf	500	2.050	5.067	92.58	0.453	0.051	-1.175	0.065	0.167	98.45	**0.020**	0.055	-2.186
rgbd	bdbf	500	**2.015**	**4.994**	**92.71**	0.392	**0.014**	**-1.215**	**0.064**	**0.166**	98.46	0.021	0.030	**-2.199**
rgb	bdbf	500	2.569	5.642	88.67	0.481	0.015	-0.979	0.098	0.199	**98.48**	0.030	**0.014**	-1.689
rgb†	log	0	6.758	11.78	61.48	1.591	0.291	2.407	**0.366**	**0.561**	**75.08**	**0.161**	0.191	**0.187**
rgb	bdbf	0	**5.809**	**9.610**	**62.78**	**1.381**	**0.264**	**0.809**	0.664	0.944	47.76	0.245	**0.044**	0.459

Table 3: Quantitative results of all methods trained and tested with 500 sparse depths and our proposed method tested with no sparse depths compared to a monocular depth prediction baseline(†). *rgbd* under the input column indicates the basis network uses the sparse depths scaffolding approach from [66], whereas *rgb* uses color image as basis network input only.

tribution. *log* and *snap+log* produce decent relative uncertainties (AUSE) but are not well-calibrated (AUCE). Figure 5 (a) (b) show sparsification error and calibration plots for the in-distribution test on both datasets, which are used to compute AUSE and AUCE respectively. We see that predictive method (*log*) performs similar to its ensemble variant *snap+log* and both are better than the pure ensemble method, *snap*. This is consistent with findings in [28, 52].

We also evaluate all methods under the effect of distributional (dataset) shift [50]. Here we mainly focus on the following two aspects: **sparsity change** and **domain shift**. For sparsity change within mid-density, we take models trained on 5% sparsity and test on varying sparsity level from 5% to 1%. Results are shown in Figure 5 (c). Note that these plots reflect how each method performs on two axes in terms of uncertainty estimation (NLL) and depth prediction (MAE), where better methods should reside closer to the lower left corner. We see that performance of all methods degrade in a similar manner with deceasing sparsity, which is largely due to the sparse depth scaffolding approach we choose. However, *bdbf* stands out with its 1% result better than the 5% results of its competitors. We refer the reader to our supplementary material for results on domain shift.

Low-density Depth Completion. In this setting, we train all methods with 500 sparse points, which is roughly 0.5% sparsity given our image size. We also introduce a slight variation of our method *bdbf(rgb)* which only uses sparse depths at the fitting stage (not as network input). Because at very low sparsity levels (*e.g.* 50 points), the scaffolding method we use for depth interpolation [66] struggles to re-
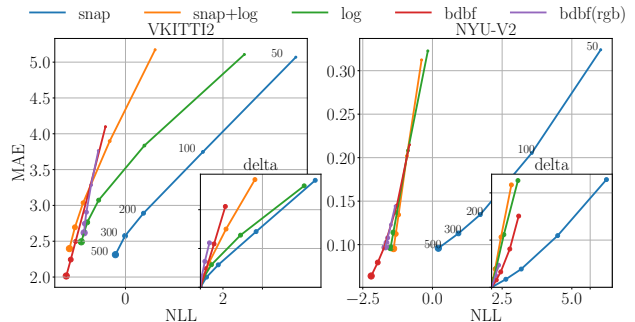

Figure 6: Sparsity change plots of all methods trained with 500 sparse depths and test with various sparsity from 500 to 50. The small subplots show how the performance of each method changes with decreasing sparsity w.r.t. the performance of their in-distribution test (500). Shorter lines indicate better sparsity-invariance.

cover the scene structure which impacts the performance of all *rgbd* methods.

The top half of Table 3 shows the in-distribution test of all methods. Among the four *rgbd* methods, *bdbf* again outperforms the rest by a large margin. *bdbf(rgb)*, despite not utilizing the rich information provided by the interpolated depths, performs on-par with the baselines. The real advantage of this approach is that it does not suffer from the artifacts caused by poor depth interpolation in the very low sparsity regime, which makes it sparsity-invariant. This claim is verified in Figure 6, which shows how each
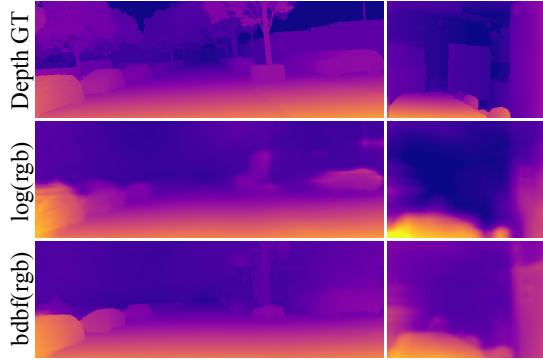
Figure 7: Qualitative results of our method tested with 0 sparse points. *log(rgb)* is trained as a monocular depth prediction network with NLL loss, which serves as a baseline. *bdbf(rgb)* is trained with 500 sparse depths.

Method	iRMSE	iMAE	RMSE	MAE
S2D [42]	2.80	1.21	814.73	249.95
Gansbeke [19]	2.19	0.93	772.87	215.02
DepthNormal[67]	2.42	1.13	777.05	235.17
DeepLiDAR [53]	2.56	1.15	758.38	226.50
FuseNet [8]	2.34	1.14	752.88	221.19
CSPN++ [9]	2.07	0.90	743.69	209.28
NLSPN [51]	1.99	0.84	741.68	199.59
GuideNet [61]	2.25	0.99	736.24	218.83
bdbf (ours)	2.37	0.89	900.38	216.44

Table 4: Comparison with selected methods on the official KITTI depth completion test set.

method's performance deteriorates with decreasing sparsity. It is shown in the small subplots that *bdbf(rgb)* is able to maintain good performance even with as few as 50 points.

Finally, we test *bdbf(rgb)* with no sparse depths, which relies only on the shared prior to make a prediction. We ignore all *rgbd* methods because with nothing to interpolate the network outputs poor solutions. We thus only compare to another baseline *log(rgb)*, which is trained for monocular depth prediction with NLL loss. Note that *bdbf(rgb)* and *log(rgb)* have exactly the same architecture (except for the last layer) and number of parameters. We see that *bdbf(rgb)* produces sharper depth than the baseline as shown in Figure 7. Quantitative results can be found in the last two rows of Table 3. The difference in performance of our method between two datasets is due to the distribution of the data: VKITTI2 contains mainly sequential driving videos, which gives a sharply peaked prior; whereas data from NYU-V2 are taken from a wide variety of scenes with different viewing angles, hence a less informative one. These results show that our learned depth bases and shared prior contain geometric information about the scene conditioned on the image and can be used under extreme conditions without catastrophic failure.

Method	MAE	RMSE	AUSE	AUCE	NLL
NCNN-L2 [14]	258.68	954.34	0.70	-	-
pNCNN [13]	283.41	1237.65	0.055	-	-
pNCNN-Exp	251.77	960.05	0.065	-	-
bdbf (ours)	206.70	876.76	0.057	0.23	-2.68

Table 5: Comparison with variations of pNCNN [13] on accuracy and uncertainty on the official KITTI validation set (with groundtruth). Note that pNCNN is *unguided*.

Further Comparisons. While our focus is on evaluating the quality of our uncertainty estimation scheme, we also evaluate depth completion performance for completeness. We trained our method with a ResNet34 encoder [25] and applied it to the KITTI depth completion benchmark with results shown in Table 4. We compare our relatively simple Bayesian filtering scheme to SOTA methods that utilize either iterative refinement [9, 51] or sub-networks with extra constraints [67, 53]. Our method compares favorably on all measures except RMSE and we observe that this difference is due to a small number of mis-attributed pixels near depth discontinuities and the use of L1 loss only. This suggests that these methods could be further improved by predicting initial depth and uncertainty estimates with our module.

We also compare with pNCNN [13] as it is the only work that provides a quantitative evaluation of predicted uncertainties for depth completion. Unfortunately, they only evaluate using a single metric, AUSE, which we argue cannot completely capture the true quality of the uncertainty estimate. Results are shown in Table 5, note that pNCNN is unguided and evaluation is done on the KITTI validation set, as groundtruth is required to compute uncertainty metrics.

5. Conclusions

In this paper, we extend Deep Basis Fitting for depth completion under a principled Bayesian framework that outputs uncertainty estimates alongside depth prediction. Compared to commonly used uncertainty estimation techniques, our integrated approach is able to produce better uncertainty estimates while being data- and compute-efficient. The benefit of being Bayesian is also demonstrated by the ability to handle very low-density sparse depths, a situation where the original DBF method struggles. Our work allows a depth completion network to be further integrated into robotics systems, where Bayesian sensor fusion is the dominant approach.

Acknowledgement

References

[1] Oisin Mac Aodha, Ahmad Humayun, M. Pollefeys, and G. Brostow. Learning a confidence measure for optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1107–1120, 2013. 6

[2] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the ekf-slam algorithm. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568, 2006. 4

[3] C. M. Bishop. Pattern recognition and machine learning (information science and statistics). 2006. 1, 3, 4

[4] Michael Bloesch, J. Czarnowski, Ronald Clark, S. Leutenegger, and A. Davison. Codeslam - learning a compact, optimisable representation for dense visual slam. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2560–2568, 2018. 1, 3

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015. 2

[6] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2, 2020. 5

[7] J. Q. Candela, C. Rasmussen, Fabian H Sinz, O. Bousquet, and B. Schölkopf. Evaluating predictive uncertainty challenge. In *MLCW*, 2005. 6

[8] Yuxiang Chen, Bin Yang, Ming Liang, and R. Urtasun. Learning joint 2d-3d representations for depth completion. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10022–10031, 2019. 2, 8

[9] Xinjing Cheng, P. Wang, Chenye Guan, and Ruigang Yang. Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. *ArXiv*, abs/1911.05377, 2020. 2, 8

[10] Nathaniel Chodosh, Chaoyang Wang, and S. Lucey. Deep convolutional compressed sensing for lidar depth completion. *ArXiv*, abs/1803.08949, 2018. 2

[11] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em - algorithm plus discussions on the paper. 1977. 4

[12] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 1, 3, 5

[13] A. Eldesokey, M. Felsberg, Karl Holmquist, and M. Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12011–12020, 2020. 3, 8

[14] A. Eldesokey, M. Felsberg, and F. Khan. Propagating confidences through cnns for sparse data regression. In *BMVC*, 2018. 3, 8

[15] David Ferstl, Christian Reinbacher, René Ranftl, M. Rüther, and H. Bischof. Image guided depth upsampling using anisotropic total generalized variation. *2013 IEEE International Conference on Computer Vision*, pages 993–1000, 2013. 2

[16] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2016. 5

[17] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, 2016. 1

[18] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 1, 2

[19] Wouter Van Gansbeke, D. Neven, Bert De Brabandere, and L. Gool. Sparse and noisy lidar completion with rgb guidance and uncertainty. *2019 16th International Conference on Machine Vision Applications (MVA)*, pages 1–6, 2019. 2, 8

[20] Jochen Gast and S. Roth. Lightweight probabilistic deep networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3369–3378, 2018. 2

[21] T. Gneiting and A. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359 – 378, 2007. 6

[22] Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011. 2

[23] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. *ArXiv*, abs/1706.04599, 2017. 1, 6

[24] Fredrik K. Gustafsson, Martin Danelljan, and Thomas Bo Schön. Evaluating scalable bayesian deep learning methods for robust computer vision. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1289–1298, 2020. 2, 3, 4, 5, 6

[25] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 8

[26] Gao Huang, Y. Li, Geoff Pleiss, Zhuang Liu, J. Hopcroft, and K. Weinberger. Snapshot ensembles: Train 1, get m for free. *ArXiv*, abs/1704.00109, 2017. 2, 4, 5, 7

[27] P. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35:492–518, 1964. 3

[28] Eddy Ilg, Özgün Çiçek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*, 2018. 2, 3, 4, 6, 7

[29] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999. 2

[30] Pasi Jylänki, Aapo Nummenmaa, and Aki Vehtari. Expectation propagation for neural networks with sparsity-promoting priors. *The Journal of Machine Learning Research*, 15(1):1849–1901, 2014. 2

[31] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*, 2017. 2, 5, 7

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 5

[33] Armen Der Kiureghian and Ove D. Ditlevsen. Aleatory or epistemic? does it matter? 2009. 2

[34] Agustinus Kristiadi, M. Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. *ArXiv*, abs/2002.10118, 2020. 1

[35] A. Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *CACM*, 2017. 5

[36] Balaji Lakshminarayanan, A. Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017. 1, 2

[37] Ang Li, Zejian Yuan, Yonggen Ling, W. Chi, Shenghao Zhang, and C. Zhang. A multi-scale guided cascade hourglass network for depth completion. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 32–40, 2020. 2

[38] J. Liu and Xiaojin Gong. Guided depth enhancement via anisotropic diffusion. In *PCM*, 2013. 2

[39] Wenxin Liu, D. Caruso, Eddy Ilg, Jing Dong, Anastasios I. Mourikis, Kostas Daniilidis, V. Kumar, and J. Engel. Tlio: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5:5653–5660, 2020. 2

[40] Antonio Loquercio, Mattia Segù, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5:3153–3160, 2020. 4

[41] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[42] Fangchang Ma, G. Cavalheiro, and S. Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. *2019 International Conference on Robotics and Automation (ICRA)*, pages 3288–3295, 2019. 5, 8

[43] Fangchang Ma and S. Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018. 2

[44] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992. 1, 2

[45] Thomas Peter Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001. 2

[46] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, CAN, 1995. AAINN02676. 2

[47] D. Nix and A. Weigend. Estimating the mean and variance of the target probability distribution. *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, 1:55–60 vol.1, 1994. 2, 3, 5, 7

[48] Sebastian Ober and C. Rasmussen. Benchmarking the neural linear model for regression. *ArXiv*, abs/1912.08416, 2019. 1, 4

[49] Manfred Opper and Ole Winther. A bayesian approach to on-line learning. *On-line learning in neural networks*, pages 363–378, 1998. 2

[50] Y. Ovadia, E. Fertig, J. Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019. 1, 6, 7

[51] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and I. S. Kweon. Non-local spatial propagation network for depth completion. *ArXiv*, abs/2007.10042, 2020. 2, 8

[52] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. *ArXiv*, abs/2005.06209, 2020. 2, 3, 4, 7

[53] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng C. Liu, Bing Zeng, and Marc Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. *ArXiv*, abs/1812.00488, 2018. 2, 8

[54] Chao Qu, Ty Nguyen, and Camillo J. Taylor. Depth completion via deep basis fitting. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 71–80, 2020. 1, 2, 3, 5, 7

[55] K. Sartipi, T. Do, Tong Ke, Khiem Vuong, and S. Roumeliotis. Deep depth estimation from visual-inertial slam. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10038–10045, 2020. 2

[56] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5

[57] Edward Snelson, C. Rasmussen, and Zoubin Ghahramani. Warped gaussian processes. In *NIPS*, 2003. 3, 5

[58] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, N. Sundaram, Md. Mostofa Ali Patwary, Prabhat, and R. Adams. Scalable bayesian optimization using deep neural networks. In *ICML*, 2015. 2, 4

[59] Niko Sünderhauf, O. Brock, W. Scheirer, Raia Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, Michael Milford, and P. Corke. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37:405 – 420, 2018. 1

[60] Chengzhou Tang and P. Tan. Ba-net: Dense bundle adjustment networks. In *ICLR*, 2019. 1

[61] J. Tang, F. Tian, W. Feng, J. Li, and Ping Tan. Learning guided convolutional network for depth completion. *ArXiv*, abs/1908.01238, 2019. 2, 8

[62] Sujay Thakur, Cooper Lorsung, Yaniv Yacoby, Finale Doshi-Velez, and Weiwei Pan. Learned uncertainty-aware (luna) bases for bayesian regression using multi-headed auxiliary networks. *ArXiv*, abs/2006.11695, 2020. 4, 5, 6

[63] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Workshop on Vision Algorithms*, 1999. 3

[64] J. Uhrig, N. Schneider, Lukas Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. *2017 International Conference on 3D Vision (3DV)*, pages 11–20, 2017. 2, 5

[65] Tsun-Hsuan Wang, Fu-En Wang, Juan-Ting Lin, Yi-Hsuan Tsai, Wei-Chen Chiu, and M. Sun. Plug-and-play: Improve depth estimation via sparse data propagation. *arXiv: Image and Video Processing*, 2018. 2

[66] Alex Wong, Xiaohan Fei, Stephanie Tsuei, and Stefano Soatto. Unsupervised depth completion from visual inertial odometry. *IEEE Robotics and Automation Letters*, 5:1899–1906, 2020. 2, 5, 7

[67] Y. Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, H. Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2811–2820, 2019. 2, 8

[68] Hongyang Xue, Shengming Zhang, and Deng Cai. Depth image inpainting: Improving low rank matrix completion with low gradient regularization. *IEEE Transactions on Image Processing*, 26:4311–4320, 2017. 2

[69] Yanchao Yang, A. Wong, and Stefano Soatto. Dense depth posterior (ddp) from single image and sparse range. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3348–3357, 2019. 2

[70] Y. Zhang and T. Funkhouser. Deep depth completion of a single rgb-d image. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 175–185, 2018. 2

[71] Shanshan Zhao, M. Gong, Huan Fu, and Dacheng Tao. Adaptive context-aware multi-modal network for depth completion. *ArXiv*, abs/2008.10833, 2020. 2

[72] Xingxing Zuo, N. Merrill, W. Li, Yiping Liu, Marc Pollefeys, and Guoquan Huang. Codevio: Visual-inertial odometry with learned optimizable dense depth. *ArXiv*, abs/2012.10133, 2020. 2