Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision

Fredrik K. Gustafsson
Department of Information Technology
Uppsala University, Sweden
fredrik.gustafsson@it.uu.se

Martin Danelljan
Computer Vision Laboratory
ETH Zurich, Switzerland
martin.danelljan@vision.ee.ethz.ch

Thomas B. Schön
Department of Information Technology
Uppsala University, Sweden
thomas.schon@it.uu.se

Abstract

While Deep Neural Networks (DNNs) have become the go-to approach in computer vision, the vast majority of these models fail to properly capture the uncertainty inherent in their predictions. Estimating this predictive uncertainty can be crucial, for instance in automotive applications. In Bayesian deep learning, predictive uncertainty is often decomposed into the distinct types of aleatoric and epistemic uncertainty. The former can be estimated by letting a DNN output the parameters of a probability distribution. Epistemic uncertainty estimation is a more challenging problem, and while different scalable methods recently have emerged, no comprehensive comparison has been performed in a real-world setting. We therefore accept this task and propose an evaluation framework for predictive uncertainty estimation that is specifically designed to test the robustness required in real-world computer vision applications. Using the proposed framework, we perform an extensive comparison of the popular ensembling and MC-dropout methods on the tasks of depth completion and street-scene semantic segmentation. Our comparison suggests that ensembling consistently provides more reliable uncertainty estimates. Code is available at `https://github.com/fregu856/evaluating_bdl`.

1 Introduction

Deep Neural Networks (DNNs) have become the standard paradigm within most computer vision problems due to their astonishing predictive power compared to previous alternatives. Current applications include many safety-critical tasks, such as street-scene semantic segmentation [10, 50, 6, 52], automotive 3D object detection [48, 44, 29, 42, 55] and depth completion [46, 33, 12]. Since erroneous predictions can have disastrous consequences, such applications require an accurate measure of the predictive uncertainty. The vast majority of the aforementioned models do however fail to properly capture the uncertainty inherent in their predictions. They are thus not fully capable of the type of *uncertainty-aware* reasoning that is highly desired in *e.g.* automotive applications.

In Bayesian deep learning, predictive uncertainty is commonly decomposed into two distinct types, which should be captured by the learned DNN [14, 25]. *Epistemic uncertainty* accounts for uncertainty in the DNN model parameters, while *aleatoric uncertainty* captures inherent and irreducible data noise. Input-dependent *aleatoric uncertainty* about the target y arises due to *e.g.* noise and ambiguities inherent in the input x. This is present in street-scene semantic segmentation, where image pixels at object boundaries are inherently ambiguous, and in 3D object detection where the location of

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.

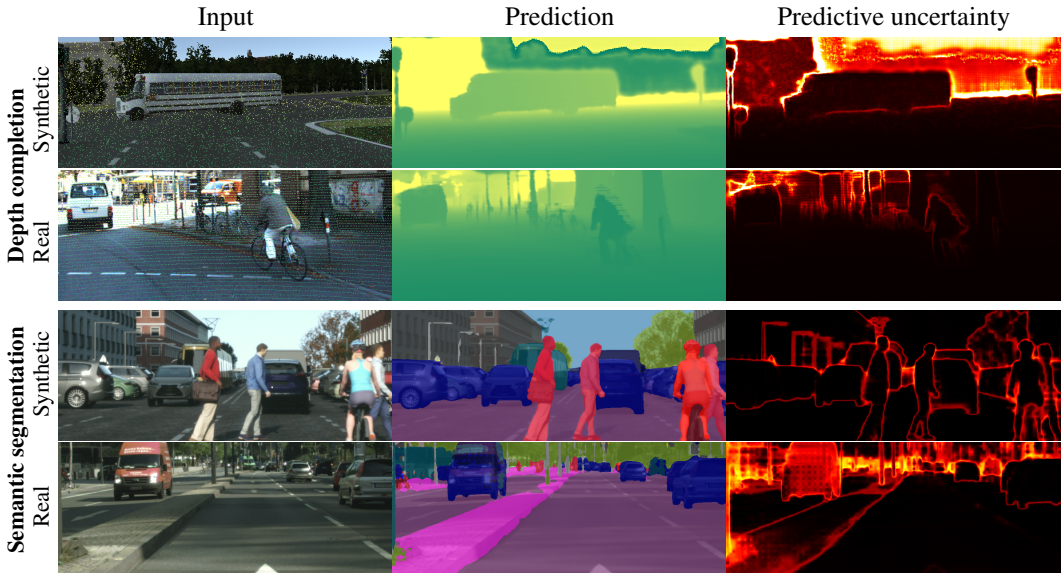


Figure 1: We evaluate predictive uncertainty estimation methods for depth completion and street-scene semantic segmentation. The DNN models have been trained exclusively on synthetic data (Virtual KITTI [13], Synscapes [50]). We here show the input (left), prediction (center) and estimated predictive uncertainty (right) for ensembling with $M = 8$, on both synthetic and real (KITTI [46], Cityscapes [10]) example validation images. Black pixels correspond to minimum uncertainty, white pixels to maximum uncertainty. The trained networks generate reasonable uncertainty estimates, even for the out-of-domain real input data.

distant objects are less certain due to noise and limited sensor resolution. In many computer vision applications, this *aleatoric uncertainty* can be effectively estimated by letting the DNN directly output the parameters of a probability distribution, modeling the conditional distribution $p(y|x)$ of the target given the input. For classification tasks, a predictive categorical distribution is commonly realized by a softmax output layer, although recent work has also explored Dirichlet models [16, 43, 35]. For regression, Laplace and Gaussian models have been employed [24, 9, 25, 28].

Directly predicting the conditional distribution $p(y|x)$ with a DNN does however not capture *epistemic uncertainty*, as information about the uncertainty in the model parameters is disregarded. This often leads to highly confident predictions that are incorrect, especially for inputs x that are not well-represented by the training distribution [19, 28]. For instance, a DNN can fail to generalize to unfamiliar weather conditions or environments in automotive applications, but still generate confident predictions. Reliable estimation of *epistemic uncertainty* is thus of great importance. However, this task has proven to be highly challenging, largely due to the vast dimensionality of the parameter space, which renders standard Bayesian inference approaches intractable. To tackle this problem, various approximations have been explored [39, 23, 4, 49, 7, 54]. Among these, MC-dropout has recently attracted considerable interest [15, 14, 25, 26, 37] due to its simplicity and scalability. For similar reasons, previous work has also explored the use of ensembling [11] as a non-Bayesian alternative [28, 9, 24]. While different scalable techniques for addressing *epistemic uncertainty* have emerged, no comprehensive comparison of such methods have been performed in a real-world setting. In this work, we therefore accept this task, aiming to benefit and inspire future research in the field.

Previous studies have provided partial insight into the performance of different scalable techniques for *epistemic uncertainty* estimation. Kendall and Gal [25] evaluated MC-dropout alone on the tasks of semantic segmentation and monocular depth regression, providing mainly qualitative results. Lakshminarayanan *et al.* [28] introduced ensembling as a non-Bayesian alternative and found it to generally outperform MC-dropout. Their experiments were however mainly based on relatively small-scale models and datasets. Ilg *et al.* [24] compared ensembling and MC-dropout on the task of optical-flow estimation in terms of their introduced AUSE metric, which is a *relative* measure of the uncertainty estimation quality. While finding ensembling to be advantageous, their experiments were limited to a fixed number ($M = 8$) of ensemble members and MC-dropout forward passes.

Contributions In this work, we study how to learn DNN models that are capable of capturing both aleatoric and epistemic uncertainty, for real-world computer vision classification and regression tasks. Following previous work, we predict the conditional distribution in order to estimate input-dependent *aleatoric uncertainty*. For *epistemic uncertainty* estimation, we extensively compare current state-of-the-art scalable alternatives: ensembling and MC-dropout. To this end, we perform a comprehensive *quantitative* evaluation of the estimated predictive uncertainty in terms of the relative AUSE metric [24] and the *absolute* measure of uncertainty calibration. To provide a deeper and more fair analysis, we study the performance as a function of the ensemble size M. Moreover, we simulate challenging real-world conditions found *e.g.* in automotive applications, where robustness to out-of-domain inputs is required to ensure safety, by training our models exclusively on synthetic data and evaluating the predictive uncertainty on real data. We also complement our real-world analysis with experiments on illustrative toy regression and classification problems. Lastly, to improve rigour of our evaluation, we repeat each experiment multiple times and report results together with the observed variation. Specifically, our main contributions are as follows: **1)** We propose an evaluation framework for predictive uncertainty estimation that is specifically designed to test the robustness required in real-world computer vision applications. **2)** We apply the proposed framework on state-of-the-art DNN models for depth completion and street-scene semantic segmentation. **3)** We perform a comprehensive comparison of ensembling and MC-dropout, the results of which suggest that ensembling consistently provides more reliable and useful predictive uncertainty estimates. Figure 1 shows example predictive uncertainty estimates generated by ensembling, and further qualitative results are available at `https://youtu.be/CabPVqtzsOI`.

2 Predictive uncertainty estimation using Bayesian deep learning

DNNs have been shown to excel at a variety of supervised machine learning problems, where the task is to predict a target value $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$. In computer vision, the input space $\mathcal{X}$ often corresponds to the space of images. For classification problems, the target space $\mathcal{Y}$ consists of a finite set of C classes, while a regression problem is characterized by a continuous target space, *e.g.* $\mathcal{Y} = \mathbb{R}^K$. For our purpose, a DNN is defined as a function $f_\theta : \mathcal{X} \to \mathcal{U}$, parameterized by $\theta \in \mathbb{R}^P$, that maps an input $x \in \mathcal{X}$ to an output $f_\theta(x) \in \mathcal{U}$. Next, we cover alternatives for estimating both the aleatoric and epistemic uncertainty in the predictions of DNN models.

Aleatoric uncertainty In classification problems, aleatoric uncertainty is commonly captured by predicting a categorical distribution $p(y|x, \theta)$. This is implemented by letting the DNN f_θ predict logit scores, which are then normalized by a Softmax function,

$$p(y|x, \theta) = \text{Cat}(y; s_\theta(x)), \qquad s_\theta(x) = \text{Softmax}(f_\theta(x)), \qquad f_\theta(x) \in \mathbb{R}^C. \tag{1}$$

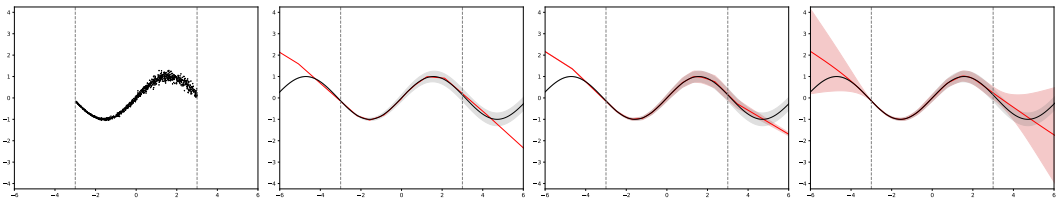
Given a training set of i.i.d. sample pairs $\mathcal{D} = \{X, Y\} = \{(x_i, y_i)\}_{i=1}^N$, $(x_i, y_i) \sim p(x, y)$, the data likelihood is obtained as $p(Y|X, \theta) = \prod_{i=1}^N p(y_i|x_i, \theta)$. The maximum-likelihood estimate of the model parameters, $\hat{\theta}_{\text{MLE}}$, is found by minimizing $-\sum_i \log p(y_i|x_i, \theta)$. For the Categorical model (1), this is equivalent to minimizing the well-known cross-entropy loss. At test time, the trained model predicts the distribution $p(y^\star|x^\star, \hat{\theta}_{\text{MLE}})$ over the target class variable $y^\star$, given a test input $x^\star$. These DNN models are thus able to capture input-dependent aleatoric uncertainty, by outputting less confident predictions for inherently ambiguous cases.

In regression problems, the most common approach is to let the DNN directly output predicted targets, $y^\star = f_{\hat{\theta}}(x^\star)$. The parameters $\hat{\theta}$ are learned by minimizing *e.g.* the L^2 or L^1 loss over the training dataset $\mathcal{D}$ [44, 29, 42]. However, such direct regression does not model aleatoric uncertainty. Instead, recent work [24, 9, 25, 28] has explored predicting the conditional distribution $p(y|x, \theta)$, similar to the classification case above. For instance, $p(y|x, \theta)$ can be parameterized by a Gaussian distribution [9, 25, 28], giving the following model in the 1D case,

$$p(y|x, \theta) = \mathcal{N}\big(y; \mu_\theta(x), \sigma_\theta^2(x)\big), \qquad f_\theta(x) = [\, \mu_\theta(x) \quad \log \sigma_\theta^2(x) \,]^\mathsf{T} \in \mathbb{R}^2. \tag{2}$$

Here, the DNN predicts the mean $\mu_\theta(x)$ and variance $\sigma_\theta^2(x)$ of the target y. The variance $\sigma_\theta^2(x)$ is naturally interpreted as a measure of input-dependent aleatoric uncertainty. Analogously to the classification case, the model parameters θ are learned by minimizing the negative log-likelihood.

Epistemic uncertainty While the above models have the potential of capturing aleatoric uncertainty, stemming from the data, they are agnostic to the uncertainty in the model parameters θ. A principled



(a) (b) (c) (d)

Figure 2: Illustrative toy regression problem. The underlying data distribution $p(y|x)$ is a Gaussian, where the mean is given by the solid black line and the variance is represented in shaded gray. The predictive mean and variance are given by the solid red line and the shaded red area, respectively. **(a)** Training dataset, containing $N = 1\,000$ examples. **(b)** A DNN trained to directly predict the target y does not capture any notion of uncertainty. **(c)** A corresponding Gaussian DNN model (2) trained via maximum-likelihood captures *aleatoric* but not epistemic uncertainty. **(d)** The Gaussian DNN model trained via approximate Bayesian inference (4) captures both *aleatoric* and *epistemic uncertainty*.

means to estimate this epistemic uncertainty is to perform Bayesian inference. The aim is to utilize the posterior distribution $p(\theta|\mathcal{D})$, which is obtained from the data likelihood and a chosen prior $p(\theta)$ by applying Bayes' theorem. The uncertainty in the model parameters is then marginalized out to obtain the predictive posterior distribution,

$$p(y^\star|x^\star, \mathcal{D}) = \int p(y^\star|x^\star, \theta)p(\theta|\mathcal{D})d\theta \approx \frac{1}{M}\sum_{i=1}^{M} p(y^\star|x^\star, \theta^{(i)}), \quad \theta^{(i)} \sim p(\theta|\mathcal{D})\,. \tag{3}$$

Here, the generally intractable integral in (3) is approximated using M Monte Carlo samples $\theta^{(i)}$, ideally drawn from the posterior. In practice however, obtaining samples from the true posterior $p(\theta|\mathcal{D})$ is virtually impossible for DNNs, requiring an approximate posterior $q(\theta) \approx p(\theta|\mathcal{D})$ to be used. We thus obtain the approximate predictive posterior as,

$$\hat{p}(y^\star|x^\star, \mathcal{D}) \triangleq \frac{1}{M}\sum_{i=1}^{M} p(y^\star|x^\star, \theta^{(i)}), \quad \theta^{(i)} \sim q(\theta)\,, \tag{4}$$

which enables us to estimate both aleatoric and epistemic uncertainty of the prediction. The quality of the approximation (4) depends on the number of samples M and the method employed for generating $q(\theta)$. Prior work on such approximate Bayesian inference methods is discussed in Section 3. For the Categorical model (1), $\hat{p}(y^\star|x^\star, \mathcal{D}) = \text{Cat}(y^\star; \hat{s}(x^\star))$, $\hat{s}(x^\star) = \frac{1}{M}\sum_{i=1}^{M} s_{\theta^{(i)}}(x^\star)$. For the Gaussian model (2), $\hat{p}(y^\star|x^\star, \mathcal{D})$ is a uniformly weighted mixture of Gaussian distributions. We approximate this mixture with a single Gaussian, see Appendix A for details.

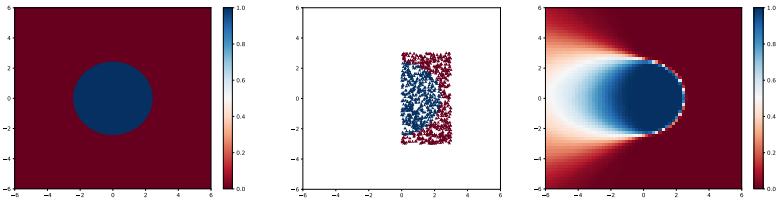
Illustrative example To visualize the problem of predictive uncertainty estimation with DNNs, we consider the problem of regressing a sinusoid corrupted by input-dependent Gaussian noise,

$$y \sim \mathcal{N}\big(\mu(x), \sigma^2(x)\big), \qquad \mu(x) = \sin(x), \qquad \sigma(x) = 0.15(1 + e^{-x})^{-1}. \tag{5}$$

Training data $\{(x_i, y_i)\}_{i=1}^{1000}$ is only given in the interval $[-3, 3]$, see Figure 2a. A DNN trained to directly predict the target y is able to accurately regress the mean for $x^\star \in [-3, 3]$, see Figure 2b. However, this model does not capture any notion of uncertainty. A corresponding Gaussian DNN model (2) trained via maximum-likelihood obtains a predictive distribution that closely matches the ground truth for $x^\star \in [-3, 3]$, see Figure 2c. While correctly accounting for aleatoric uncertainty, this model generates overly confident predictions for inputs $|x^\star| > 3$ not seen during training. Finally, the Gaussian DNN model trained via approximate Bayesian inference (4), with a prior $p(\theta) = \mathcal{N}(0, I_P)$ and $M = 1\,000$ samples obtained via Hamiltonian Monte Carlo [40], is additionally able to predict more reasonable uncertainties in the region where no training data was available, see Figure 2d.

3 Related work

In this section, we discuss prior work on approximate Bayesian inference and ensembling. We also note that ensembling naturally can be viewed as an approximate Bayesian inference method.



(a) (b) (c)

Figure 3: Toy binary classification problem. **(a)** True data generator, red and blue represents the two classes. **(b)** Training dataset. **(c)** "Ground truth" predictive distribution, obtained using HMC.

Approximate Bayesian inference The method for approximating the posterior $q(\theta) \approx p(\theta|\mathcal{D}) = p(Y|X,\theta)p(\theta)/p(Y|X)$ is a crucial choice, determining the quality of the approximate predictive posterior $\hat{p}(y^\star|x^\star,\mathcal{D})$ in (4). There exists two main paradigms for constructing $q(\theta)$, the first one being *Markov chain Monte Carlo (MCMC)* methods. Here, samples $\theta^{(i)}$ approximately distributed according to the posterior are obtained by simulating a Markov chain with $p(\theta|\mathcal{D})$ as its stationary distribution. For DNNs, this approach was pioneered by Neal [39], who employed Hamiltonian Monte Carlo (HMC) [40] on small feed-forward neural networks. HMC entails performing Metropolis-Hastings [36, 21] updates using Hamiltonian dynamics based on the potential energy $U(\theta) \triangleq -\log p(Y|X,\theta)p(\theta)$. To date, it is considered a "gold standard" method for approximate Bayesian inference, but does not scale to large DNNs or large-scale datasets. Therefore, *Stochastic Gradient MCMC (SG-MCMC)* [34] methods have been explored, in which stochastic gradients are utilized in place of their full-data counterparts. SG-MCMC variants include Stochastic Gradient Langevin Dynamics (SGLD) [49], where samples $\theta^{(i)}$ are collected from the parameter trajectory given by the update equation $\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \widetilde{U}(\theta_t) + \sqrt{2\alpha_t}\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0,1)$ and $\nabla_\theta \widetilde{U}(\theta)$ is the stochastic gradient of $U(\theta)$. Save for the noise term $\sqrt{2\alpha_t}\epsilon_t$, this update is identical to the conventional SGD update when minimizing the maximum-a-posteriori (MAP) objective $-\log p(Y|X,\theta)p(\theta)$. Similarly, Stochastic Gradient HMC (SGHMC) [7] corresponds to SGD with momentum injected with properly scaled noise. Given a limited computational budget, SG-MCMC methods can however struggle to explore the high-dimensional and highly multi-modal posteriors of large DNNs. To mitigate this problem, Zhang *et al.* [54] proposed to use a cyclical stepsize schedule to help escaping local modes.

The second paradigm is that of *Variational Inference (VI)* [23, 2, 18, 4]. Here, a distribution $q_\phi(\theta)$ parameterized by variational parameters ϕ is explicitly chosen, and the best possible approximation is found by minimizing the Kullback-Leibler (KL) divergence with respect to the true posterior $p(\theta|\mathcal{D})$. While principled, VI methods generally require sophisticated implementations, especially for more expressive variational distributions $q_\phi(\theta)$ [31, 32, 53]. A particularly simple and scalable variant is MC-dropout [15]. The method entails using dropout [45] also at test time, which can be interpreted as performing VI with a Bernoulli variational distribution [15, 26, 37]. The approximate predictive posterior (4) is obtained by performing M stochastic forward passes on the same input.

Ensembling Lakshminarayanan *et al.* [28] created a parametric model $p(y|x,\theta)$ of the conditional distribution using a DNN f_θ, and learned multiple point estimates $\{\hat{\theta}^{(m)}\}_{m=1}^M$ by repeatedly minimizing the MLE objective $-\log p(Y|X,\theta)$ with random initialization. They then averaged over the corresponding parametric models to obtain the predictive distribution,

$$\hat{p}(y^\star|x^\star) \triangleq \frac{1}{M}\sum_{m=1}^{M} p(y^\star|x^\star,\hat{\theta}^{(m)}). \tag{6}$$

The authors considered this a non-Bayesian alternative to predictive uncertainty estimation. However, since $\{\hat{\theta}^{(m)}\}_{m=1}^M$ always can be seen as samples from some distribution $\hat{q}(\theta)$, we note that (6) is virtually identical to (4). Ensembling can thus also be viewed as approximate Bayesian inference, where the level of approximation is determined by how well the implicit sampling distribution $\hat{q}(\theta)$ approximates the posterior $p(\theta|\mathcal{D})$. Ideally, we want $\{\hat{\theta}^{(m)}\}_{m=1}^M$ to be distributed exactly according to $p(\theta|\mathcal{D}) \propto p(Y|X,\theta)p(\theta)$. Since $p(Y|X,\theta)$ is highly *multi-modal* in the parameter space for DNNs [1, 8], so is $p(\theta|\mathcal{D})$. By minimizing $-\log p(Y|X,\theta)$ multiple times, starting from *randomly chosen* initial points, we are likely to end up in different local optima. Ensembling can thus generate a compact set of samples $\{\hat{\theta}^{(m)}\}_{m=1}^M$ that, even for small values of M, captures this important aspect of multi-modality in $p(\theta|\mathcal{D})$.

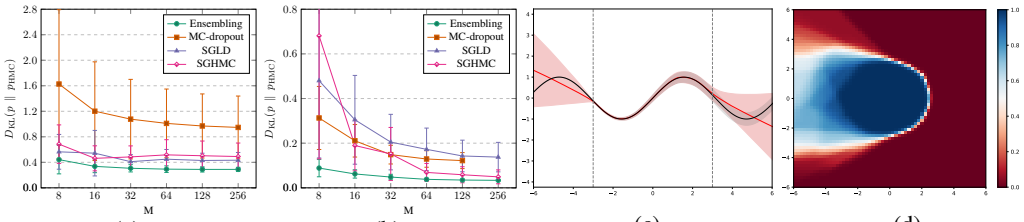


Figure 4: Results for the illustrative toy problems. **(a)** Quantitative results for the regression case. **(b)** Quantitative results for the classification case. The two plots show the KL divergence between the predictive distribution estimated by each method and the HMC "ground truth", for different number of samples M. **(c)** An example predictive distribution for ensembling with $M = 16$ in the regression case. **(d)** An example predictive distribution for ensembling with $M = 16$ in the classification case.

4 Experiments

We conduct experiments both on illustrative toy regression and classification problems (Section 4.1), and on the real-world computer vision tasks of depth completion (Section 4.2) and street-scene semantic segmentation (Section 4.3). Our evaluation is motivated by real-world conditions found *e.g.* in automotive applications, where robustness to varying environments and weather conditions is required to ensure safety. Since images captured in these different circumstances could all represent distinctly different regions of the vast input image space, it is infeasible to ensure that all encountered inputs will be well-represented by the training data. Thus, we argue that robustness to out-of-domain inputs is crucial in such applications. To simulate these challenging conditions and test the robustness required for such real-world scenarios, we train all models on synthetic data and evaluate them on real-world data. To improve rigour of our evaluation, we repeat each experiment multiple times and report results together with the observed variation. A more detailed description of all results are found in the Appendix (Appendix B.3, C.2, D.2). All experiments are implemented in PyTorch [41].

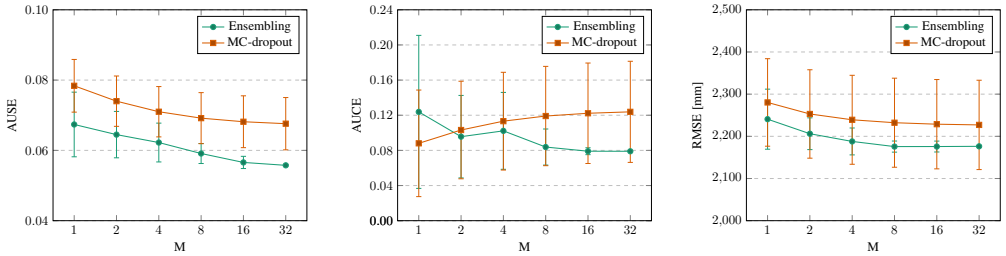
4.1 Illustrative toy problems

We first present results on illustrative toy problems to gain insights into how ensembling and MC-dropout fare against other approximate Bayesian inference methods. For regression, we conduct experiments on the 1D problem defined in (5) and visualized in Figure 2. We use the Gaussian model (2) with two separate feed-forward neural networks outputting $\mu_\theta(x)$ and $\log \sigma_\theta^2(x)$. We evaluate the methods by quantitatively measuring how well the obtained predictive distributions approximate that of the "gold standard" HMC with $M = 1\,000$ samples and prior $p(\theta) = \mathcal{N}(0, I_P)$. We thus consider the predictive distribution visualized in Figure 2d ground truth, and take as our metric the KL divergence $D_{\mathrm{KL}}(p \parallel p_{\mathrm{HMC}})$ with respect to this target distribution p_{HMC}. For classification, we conduct experiments on the binary classification problem in Figure 3. The true data generator is visualized in Figure 3a, where red and blue represents the two classes. The training dataset contains 520 examples of each class, and is visualized in Figure 3b. We use the Categorical model (1) with a feed-forward neural network. As for regression, we quantitatively measure how well the obtained predictive distributions approximate that of HMC, which is visualized in Figure 3c. A more detailed description of the experimental setup is found in Appendix B.1, and implementation details in Appendix B.2.

Results A quantitative comparison of ensembling, MC-dropout, SGLD and SGHMC in terms of $D_{\mathrm{KL}}(p \parallel p_{\mathrm{HMC}})$ is found in Figure 4a, 4b. The Adam optimizer [27] is here used for both ensembling and MC-dropout. We observe that ensembling consistently outperforms the compared methods, and MC-dropout in particular. This result is qualitatively supported by visualized predictive distributions found in Appendix B.5. Example predictive distributions for ensembling with $M = 16$ are shown in Figure 4c, 4d. We observe that ensembling can provide reasonable approximations to HMC, even for relatively small values of M.

4.2 Depth completion

In depth completion, we are given an RGB image $x_{\mathrm{img}} \in \mathbb{R}^{h \times w \times 3}$ from a forward-facing camera and an associated *sparse* depth map $x_{\mathrm{sparse}} \in \mathbb{R}^{h \times w}$. Only non-zero pixels of x_{sparse} correspond



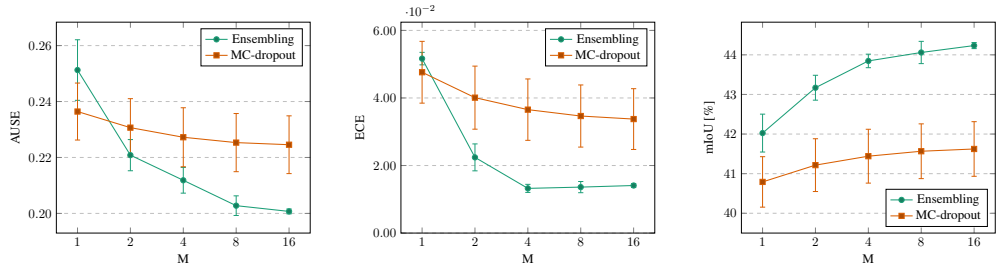
(a) AUSE. (b) AUCE. (c) RMSE.

Figure 5: Quantitative results for depth completion. The plots show a comparison of ensembling and MC-dropout in terms of AUSE **(a)**, AUCE **(b)** and RMSE **(c)** on the KITTI depth completion validation dataset, for different number of samples M. Lower is better for all three metrics.

to LiDAR depth measurements, projected onto the image plane. The goal is to predict a dense depth map $y \in \mathbb{R}^{h \times w}$ of the scene. We utilize the KITTI depth completion [17, 46] and Virtual KITTI [13] datasets. KITTI depth completion contains more than $80\,000$ RGB images x_{img}, sparse depth maps x_{sparse} and semi-dense target maps y. There are $1\,000$ selected validation examples, which we use for evaluation. Only about 4% of the pixels in x_{sparse} are non-zero and thus correspond to depth measurements. The semi-dense target maps are created by merging the LiDAR scans from 11 consecutive frames into one, producing y in which roughly 30% of the pixels are non-zero. Virtual KITTI contains synthetic RGB images x_{img} and dense depth maps x_{dense} extracted from 5 sequences in a virtual world. It contains $2\,126$ unique frames, of which there are 10 different versions corresponding to various simulated weather and lighting conditions. We take sequence 0002 as our validation set, leaving a total of $18\,930$ training examples. We create targets y for training by setting all pixels in x_{dense} corresponding to a depth > 80m to 0, and then also randomly sample 5% of the remaining non-zero pixels uniformly to create x_{sparse}. We use the DNN model presented by Ma *et al.* [33]. The inputs x_{img}, x_{sparse} are separately processed by initial convolutional layers, concatenated and fed to an encoder-decoder architecture based on ResNet34 [22]. We employ the Gaussian model (2) by duplicating the final convolutional layer, outputting $\mu \in \mathbb{R}^{h \times w}$ and $\log \sigma^2 \in \mathbb{R}^{h \times w}$ instead of the plain depth $\hat{y} \in \mathbb{R}^{h \times w}$. We also employ the same basic training procedure as Ma *et al.* to train all our models, see Appendix C.1 for details. For the MC-dropout comparison, we take inspiration from [26] and place a dropout layer with $p = 0.5$ after the three last encoder blocks and the four first decoder blocks.

Evaluation metrics We evaluate the methods in terms of the *Area Under the Sparsification Error curve (AUSE)* metric, as introduced by Ilg *et al.* [24]. AUSE is a *relative* measure of the uncertainty estimation quality, comparing the ordering of predictions induced by the estimated predictive uncertainty (sorted from least to most uncertain) with the "oracle" ordering in terms of the true prediction error. We compute AUSE in terms of *Root Mean Squared Error (RMSE)* and based on all pixels in the entire evaluation dataset. A perfect AUSE score can be achieved even if the true predictive uncertainty is consistently underestimated. As an *absolute* measure of uncertainty estimation quality, we therefor also evaluate the methods in terms of calibration [5, 47]. Since our models output the mean and variance of a Gaussian distribution for each pixel, we can construct symmetric prediction intervals of varying confidence level $p \in]0, 1[$ using the corresponding quantiles. When computing the proportion of pixels for which the prediction interval covers the target, we expect this value to equal $p \in]0, 1[$ for a perfectly calibrated model. We compute the absolute error with respect to perfect calibration for 100 values of $p \in]0, 1[$ and use the area under this curve as our metric, which we call *Area Under the Calibration Error curve (AUCE)*. We also evaluate in terms of the RMSE metric.

Results A comparison of ensembling and MC-dropout in terms of AUSE, AUCE and RMSE on the KITTI depth completion validation dataset is found in Figure 5. We observe in Figure 5a that ensembling consistently outperforms MC-dropout in terms of AUSE. However, the curves decrease as a function of M in a similar manner. Sparsification plots and sparsification error curves are found in Appendix C.3. A ranking of the methods can be more readily conducted based on Figure 5b, where we observe a clearly improving trend as M increases for ensembling, whereas MC-dropout gets progressively worse. This result is qualitatively supported by the calibration plots found in Appendix C.3. Note that $M = 1$ corresponds to the baseline of only estimating aleatoric uncertainty.



(a) AUSE. (b) ECE. (c) mIoU.

Figure 6: Quantitative results for street-scene semantic segmentation. The plots show a comparison of ensembling and MC-dropout in terms of AUSE (lower is better) **(a)**, ECE (lower is better) **(b)** and mIoU (higher is better) **(c)** on the Cityscapes validation dataset, for different number of samples M.

4.3 Street-scene semantic segmentation

In this task, we are given an RGB image $x \in \mathbb{R}^{h \times w \times 3}$ from a forward-facing camera. The goal is to predict y of size $h \times w$, in which each pixel is assigned to one of C different class labels (road, sidewalk, car, etc.). We utilize the popular Cityscapes [10] and recent Synscapes [50] datasets. Cityscapes contains $5\,000$ finely annotated images, mainly collected in various German cities. The annotations includes 30 class labels, but only $C = 19$ are used in the training of models. Its validation set contains 500 examples, which we use for evaluation. Synscapes contains $25\,000$ synthetic images, all captured in virtual urban environments. To match the size of Cityscapes, we randomly select $2\,975$ of these for training and 500 for validation. The images are annotated with the same class labels as Cityscapes. We use the DeepLabv3 DNN model presented by Chen *et al.* [6]. The input image x is processed by a ResNet101 [22], outputting a feature map of stride 8. The feature map is further processed by an ASPP module and a 1×1 convolutional layer, outputting logits at 1/8 of the original resolution. These are then upsampled to image resolution using bilinear interpolation. The conventional Categorical model (1) is thus used for each pixel. We base our implementation on the one by Yuan and Wang [52], and also follow the same basic training procedure, see Appendix D.1 for details. For reference, the model obtains an mIoU [30] of 76.04% when trained on Cityscapes and evaluated on its validation set. For the MC-dropout comparison, we take inspiration from [37] and place a dropout layer with $p = 0.5$ after the four last ResNet blocks.

Evaluation metrics As for depth completion, we evaluate the methods in terms of the AUSE metric. In this classification setting, we compare the "oracle" ordering with the one induced by the predictive entropy. We compute AUSE in terms of Brier score and based on all pixels in the evaluation dataset. We also evaluate in terms of calibration by the *Expected Calibration Error (ECE)* metric [19, 38]. All predictions are here partitioned into L bins based on the maximum assigned confidence. For each bin, the difference between the average predicted confidence and the actual accuracy is then computed, and ECE is obtained as the weighted average of these differences. We use $L = 10$ bins.

Results A comparison of ensembling and MC-dropout in terms of AUSE, ECE and mIoU on the Cityscapes validation dataset is found in Figure 6. We observe that the metrics clearly improve as functions of M for both ensembling and MC-dropout, demonstrating the importance of epistemic uncertainty estimation. The rate of improvement is generally greater for ensembling. For ECE, we observe in Figure 6b a drastic improvement for ensembling as M is increased, followed by a distinct plateau. According to the condensed reliability diagrams in Appendix D.3, this corresponds to a transition from clear model over-confidence to slight over-conservatism. For MC-dropout, the corresponding diagrams suggest a stagnation while the model still is somewhat over-confident.

5 Discussion & conclusion

We proposed an evaluation framework for predictive uncertainty estimation that is specifically designed to test the robustness required in real-world computer vision applications. Using the proposed framework, we performed a comprehensive comparison of ensembling and MC-dropout, the results of which suggest that ensembling consistently provides more reliable uncertainty estimates. We attribute the success of ensembling to its ability, due to the random initialization, to capture the important aspect of multi-modality present in the posterior distribution of DNNs. MC-dropout has a

large design-space compared to ensembling, and while careful tuning of MC-dropout potentially could close the performance gap on individual tasks, the simplicity and general applicability of ensembling must be considered key strengths. The main drawback of both methods is the computational cost at test time that grows linearly with M, limiting real-time applicability. Here, future work includes exploring the effect of model pruning techniques [51, 20] on predictive uncertainty quality. For ensembling, sharing early stages of the DNN among ensemble members is also an interesting future direction. A weakness of ensembling is the additional training required, which also scales linearly with M. The training of different ensemble members can however be performed in parallel, making it less of an issue in practice given appropriate computing infrastructure. In conclusion, our comparison suggests that ensembling should be considered a go-to method for scalable Bayesian deep learning.

Acknowledgments

This research was financially supported by the Swedish Foundation for Strategic Research (SSF) via the project *ASSEMBLE* (contract number: RIT15-0012) and by the project *Learning flexible models for nonlinear dynamics* (contract number: 2017-03807), funded by the Swedish Research Council. We also thank Juozas Vaicenavicius for his helpful feedback.

References

[1] P. Auer, M. Herbster, and M. K. Warmuth. Exponentially many local minima for single neurons. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 316–322, 1996.

[2] D. Barber and C. M. Bishop. Ensemble learning in Bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238, 1998.

[3] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

[4] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning (ICML)*, pages 1613–1622, 2015.

[5] J. Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society*, 135(643):1512–1519, 2009.

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

[7] T. Chen, E. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning (ICML)*, pages 1683–1691, 2014.

[8] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[9] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4759–4770, 2018.

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

[11] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[12] A. Eldesokey, M. Felsberg, and F. S. Khan. Confidence propagation through cnns for guided sparse depth regression. *arXiv preprint arXiv:1811.01791*, 2018.

[13] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[15] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016.

[16] J. Gast and S. Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3369–3378, 2018.

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

[18] A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2348–2356, 2011.

[19] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1321–1330, 2017.

[20] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[21] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[23] G. Hinton and D. Van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory (COLT)*, 1993.

[24] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Bro. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018.

[25] A. Kendall and Y. Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5574–5584, 2017.

[26] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6402–6413, 2017.

[29] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[31] C. Louizos and M. Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1708–1716, 2016.

[32] C. Louizos and M. Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2218–2227, 2017.

[33] F. Ma, G. V. Cavalheiro, and S. Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[34] Y.-A. Ma, T. Chen, and E. Fox. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2917–2925, 2015.

[35] A. Malinin and M. Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7047–7058, 2018.

[36] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6): 1087–1092, 1953.

[37] J. Mukhoti and Y. Gal. Evaluating Bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018.

[38] M. P. Naeini, G. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[39] R. M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[40] R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2: 113–162, 2011.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NeurIPS - Autodiff Workshop*, 2017.

[42] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D object detection from RGB-D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 918–927, 2018.

[43] M. Sensoy, L. Kaplan, and M. Kandemir. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3179–3189, 2018.

[44] S. Shi, X. Wang, and H. Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[46] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant CNNs. In *International Conference on 3D Vision (3DV)*, 2017.

[47] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. B. Schön. Evaluating model calibration in classification. *arXiv preprint arXiv:1902.06977*, 2019.

[48] Z. Wang and K. Jia. Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection. *arXiv preprint arXiv:1903.01864*, 2019.

[49] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning (ICML)*, pages 681–688, 2011.

[50] M. Wrenninge and J. Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705*, 2018.

[51] T.-J. Yang, Y.-H. Chen, and V. Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5687–5695, 2017.

[52] Y. Yuan and J. Wang. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018.

[53] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse. Noisy natural gradient as variational inference. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5847–5856, 2018.

[54] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson. Cyclical stochastic gradient MCMC for Bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.

[55] Y. Zhou and O. Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018.

Appendix A Approximating a mixture of Gaussian distributions

For the Gaussian model (2), $\hat{p}(y^\star|x^\star, \mathcal{D})$ in (4) is a uniformly weighted mixture of Gaussian distributions. We approximate this mixture with a single Gaussian parameterized by the mixture mean and variance:

$$\hat{p}(y^\star|x^\star, \mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M} p(y^\star|x^\star, \theta^{(i)}), \quad \theta^{(i)} \sim q(\theta),$$

$$\hat{p}(y^\star|x^\star, \mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{N}(y^\star; \mu_{\theta^{(i)}}(x^\star),\, \sigma^2_{\theta^{(i)}}(x^\star)), \quad \theta^{(i)} \sim q(\theta),$$

$$\hat{p}(y^\star|x^\star, \mathcal{D}) \approx \mathcal{N}(y^\star; \hat{\mu}(x^\star),\, \hat{\sigma}^2(x^\star)),$$

$$\hat{\mu}(x) = \frac{1}{M}\sum_{i=1}^{M} \mu_{\theta^{(i)}}(x), \quad \hat{\sigma}^2(x) = \frac{1}{M}\sum_{i=1}^{M}\left(\left(\mu_{\theta^{(i)}}(x) - \hat{\mu}(x)\right)^2 + \sigma^2_{\theta^{(i)}}(x)\right), \quad \theta^{(i)} \sim q(\theta).$$

Appendix B Illustrative toy problems

In this appendix, further details on the illustrative toy problems experiments (Section 4.1) are provided.

B.1 Experimental setup

Figure 4a (regression) shows $D_{\mathrm{KL}}(p \parallel p_{\mathrm{HMC}})$ computed on $[-7, 7]$. All training data was given in $[-3, 3]$.

Figure 4b (classification) shows $D_{\mathrm{KL}}(p \parallel p_{\mathrm{HMC}})$ computed on the region $-6 \leq x_1 \leq 6$, $-6 \leq x_2 \leq 6$. All training data was given in the region $0 \leq x_1 \leq 3$, $-3 \leq x_2 \leq 3$.

For regression, $D_{\mathrm{KL}}(p \parallel p_{\mathrm{HMC}})$ is computed using the formula for KL divergence between two Gaussian distributions $p_1(x) = \mathcal{N}(x; \mu_1, \sigma_1^2)$, $p_2(x) = \mathcal{N}(x; \mu_2, \sigma_2^2)$:

$$D_{\mathrm{KL}}(p_1 \parallel p_2) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}.$$

For classification, $D_{\mathrm{KL}}(p \parallel p_{\mathrm{HMC}})$ is computed using the formula for KL divergence between two discrete distributions $q_1(x)$, $q_2(x)$:

$$D_{\mathrm{KL}}(q_1 \parallel q_2) = \sum_{x \in \mathcal{X}} q_1(x) \log\frac{q_1(x)}{q_2(x)}.$$

For both regression and classification, HMC with prior $p(\theta) = \mathcal{N}(0, I_P)$ and $M = 1\,000$ samples is implemented using Pyro [3]. Specifically, we use pyro.infer.mcmc.MCMC with pyro.infer.mcmc.NUTS as kernel, num_samples $= 1\,000$ and warmup_steps $= 1\,000$.

B.2 Implementation details

For regression, we use the Gaussian model (2) with two separate feed-forward neural networks outputting $\mu_\theta(x)$ and $\log\sigma^2_\theta(x)$. Both neural networks have 2 hidden layers of size 10.

For classification, we use the Categorical model (1) with a feed-forward neural network with 2 hidden layers of size 10.

For the MC-dropout comparison, we place a dropout layer after the first hidden layer of each neural network. For regression, we use a drop probability $p = 0.2$. For classification, we use $p = 0.1$.

For ensembling, we train all ensemble models for 150 epochs with the Adam optimizer, a batch size of 32 and a fixed learning rate of 0.001.

For MC-dropout, we train models for 300 epochs with the Adam optimizer, a batch size of 32 and a fixed learning rate of 0.001.

For ensembling and MC-dropout, we minimize the MAP objective $-\log p(Y|X,\theta)p(\theta)$. In our case where the model parameters $\theta \in \mathbb{R}^P$ and $p(\theta) = \mathcal{N}(0, I_P)$, this corresponds to the following loss for regression:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{(y_i - \hat{\mu}(x_i))^2}{\hat{\sigma}^2(x_i)} + \log \hat{\sigma}^2(x_i) + \frac{1}{N}\theta^\mathsf{T}\theta.$$

For classification, where $y_i = [\, y_{i,1} \dots y_{i,C} \,]^\mathsf{T}$ (one-hot encoded) and $\hat{s}(x_i) = [\, \hat{s}(x_i)_1 \dots \hat{s}(x_i)_C \,]^\mathsf{T}$ is the Softmax output, it corresponds to the following loss:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} y_{i,k} \log \hat{s}(x_i)_k + \frac{1}{2N}\theta^\mathsf{T}\theta.$$

For SGLD, we extract samples from the parameter trajectory given by the update equation:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \widetilde{U}(\theta_t) + \sqrt{2\alpha_t}\epsilon_t,$$

where $\epsilon_t \sim \mathcal{N}(0,1)$, $\nabla_\theta \widetilde{U}(\theta)$ is the stochastic gradient of $U(\theta) = -\log p(Y|X,\theta)p(\theta)$ and α_t is the stepsize. We run it for a total number of steps corresponding to $256 \cdot 150$ epochs with a batch size of 32. The stepsize α_t is decayed according to:

$$\alpha_t = \alpha_0(1 - \frac{t}{T})^{0.9}, \quad t = 1, 2, \dots, T,$$

where T is the total number of steps, $\alpha_0 = 0.01$ (the initial stepsize) for regression and $\alpha_0 = 0.05$ for classification. $M \in \{8, 16, 32, 64, 128, 256\}$ samples are extracted starting at step $t = \text{int}(0.75T)$, ending at step $t = T$ and spread out evenly between.

For SGHMC, we extract samples from the parameter trajectory given by the update equation:

$$\begin{aligned}\theta_{t+1} &= \theta_t + r_t, \\ r_{t+1} &= (1-\eta)r_t - \alpha_t \nabla_\theta \widetilde{U}(\theta_t) + \sqrt{2\eta\alpha_t}\epsilon_t,\end{aligned}$$

where $\epsilon_t \sim \mathcal{N}(0,1)$, $\nabla_\theta \widetilde{U}(\theta)$ is the stochastic gradient of $U(\theta) = -\log p(Y|X,\theta)p(\theta)$, α_t is the stepsize and $\eta = 0.1$. We run it for a total number of steps corresponding to $256 \cdot 150$ epochs with a batch size of 32. The stepsize α_t is decayed according to:

$$\alpha_t = \alpha_0(1 - \frac{t}{T})^{0.9}, \quad t = 1, 2, \dots, T,$$

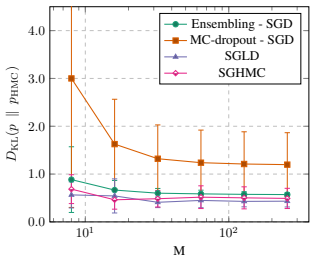
where T is the total number of steps, $\alpha_0 = 0.001$ (the initial stepsize) for regression and $\alpha_0 = 0.01$ for classification. $M \in \{8, 16, 32, 64, 128, 256\}$ samples are extracted starting at step $t = \text{int}(0.75T)$, ending at step $t = T$ and spread out evenly between.

For all models, we randomly initialize the parameters θ using the default initializer in PyTorch.

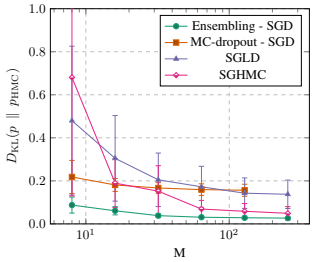
B.3 Description of results

The results in Figure 4a, 4b were obtained in the following way:

- **Ensembling:** 1024 models were trained using the same training procedure, the mean and standard deviation was computed based on $1024/M$ unique sets of models for $M \in \{8, 16, 32, 64, 128, 256\}$.
- **MC-dropout:** 10 models were trained using the same training procedure, based on which the mean and standard deviation was computed.
- **SGLD:** 6 models were trained using the same training procedure, based on which the mean and standard deviation was computed.
- **SGHMC:** 6 models were trained using the same training procedure, based on which the mean and standard deviation was computed.

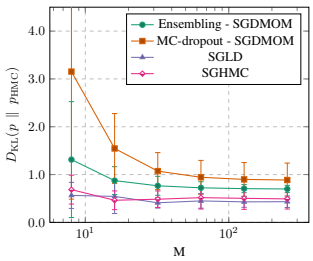


(a) Regression

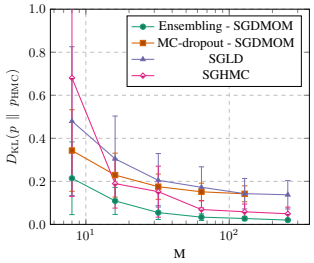


(b) Classification

Figure 7: Illustrative toy problems, quantitative results. SGD is used for ensembling and MC-dropout instead of Adam.



(a) Regression



(b) Classification

Figure 8: Illustrative toy problems, quantitative results. SGD with momentum is used for ensembling and MC-dropout instead of Adam.

B.4 Additional results

Figure 7 and Figure 8 show the same comparison as Figure 4a, 4b, but using SGD and SGD with momentum for ensembling and MC-dropout, respectively. We observe that ensembling consistently outperforms the compared methods for classification, but that SGLD and SGHMC has better performance for regression in these cases. SGLD and SGHMC are however trained for 256 times longer than each ensemble model, complicating the comparison somewhat. If SGLD and SGHMC instead are trained for just 64 times longer than each ensemble model, we observe in Figure 9 that they are consistently outperformed by ensembling.

For MC-dropout using Adam, we also varied the drop probability p and chose the best performing variant. These results are found in Figure 10, in which * marks the chosen variant.

B.5 Qualitative results

Here, we show visualizations of predictive distributions obtained by the different methods. Figure 11, 15 for ensembling, Figure 12, 16 for MC-dropout, Figure 13, 17 for SGLD, and Figure 14, 18 for SGHMC.

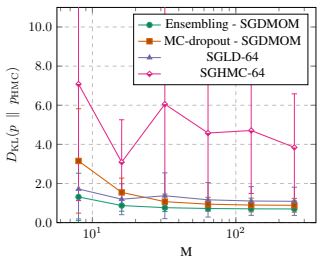
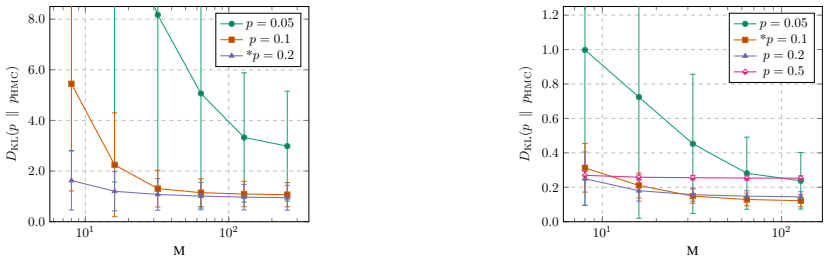


Figure 9: Illustrative toy regression problem, quantitative results. SGD with momentum is used for ensembling and MC-dropout instead of Adam. Less training for SGLD and SGHMC.



(a) Regression (b) Classification

Figure 10: Illustrative toy problems, quantitative results. MC-dropout using Adam.

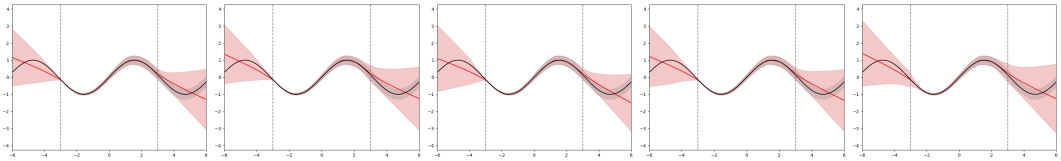


Figure 11: Toy regression problem, ensembling, $M = 64$. Examples of predictive distributions.

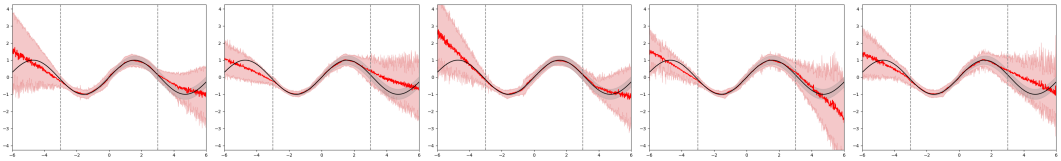


Figure 12: Toy regression problem, MC-dropout, $M = 64$. Examples of predictive distributions.

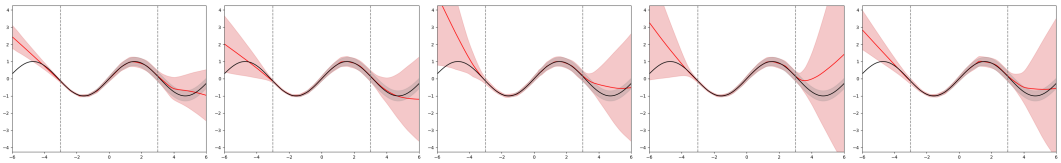


Figure 13: Toy regression problem, SGLD, $M = 64$. Examples of predictive distributions.

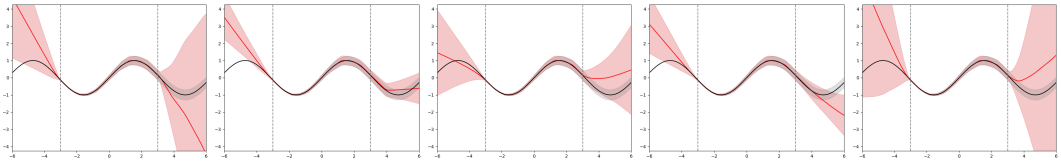


Figure 14: Toy regression problem, SGHMC, $M = 64$. Examples of predictive distributions.

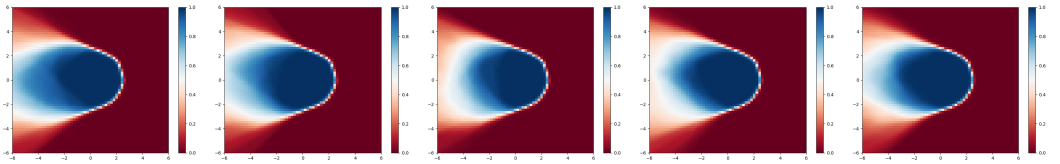


Figure 15: Toy classification problem, ensembling, $M = 64$. Examples of predictive distributions.

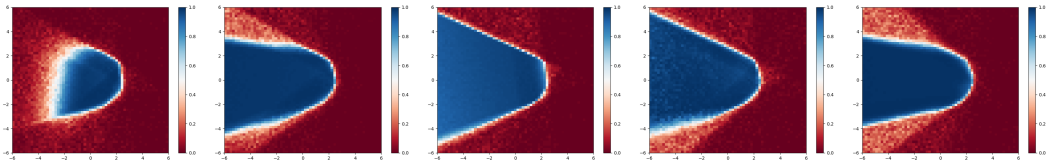


Figure 16: Toy classification problem, MC-dropout, $M = 64$. Examples of predictive distributions.

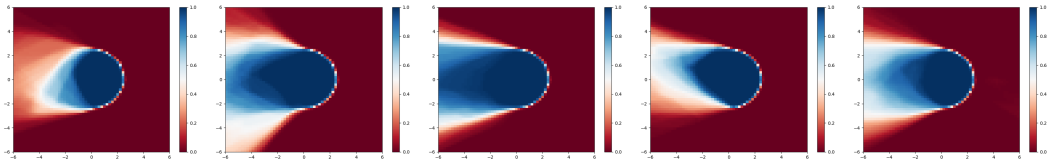


Figure 17: Toy classification problem, SGLD, $M = 64$. Examples of predictive distributions.

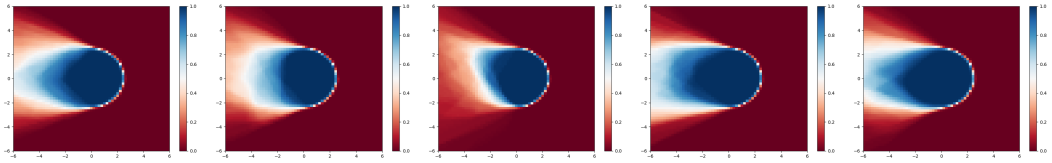


Figure 18: Toy classification problem, SGHMC, $M = 64$. Examples of predictive distributions.

Appendix C Depth completion

In this appendix, further details on the depth completion experiments (Section 4.2) are provided.

C.1 Training details

For both ensembling and MC-dropout, we train all models for $40\,000$ steps with the Adam optimizer, a batch size of 4, a fixed learning rate of 10^{-5} and weight decay of 0.0005. We use a smaller batch size and train for fewer steps than Ma *et al.* [33] to enable an extensive evaluation with repeated experiments. For the same reason, we also train on randomly selected image crops of size 352×352. The only other data augmentation used is random flipping along the vertical axis. We follow Ma *et al.* and randomly initialize all network weights from $\mathcal{N}(0, 10^{-3})$ and all network biases with 0s. Models are trained on a single NVIDIA TITAN Xp GPU with 12GB of RAM.

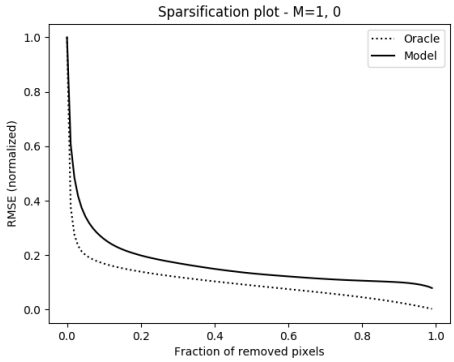
C.2 Description of results

The results in Figure 5 (Section 4.2) were obtained in the following way:

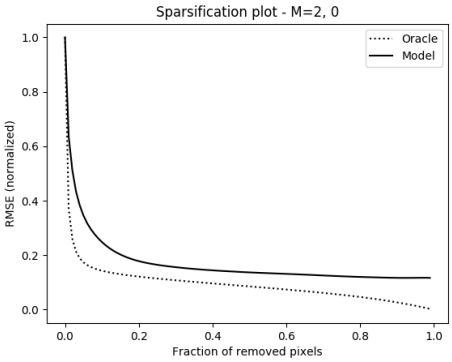
- **Ensembling:** 33 models were trained using the same training procedure, the mean and standard deviation was computed based on 32 ($M = 1$), 16 ($M = 2, 4, 8, 16$) or 4 ($M = 32$) sets of randomly drawn models. The same set could not be drawn more than once.
- **MC-dropout:** 16 models were trained using the same training procedure, based on which the mean and standard deviation was computed.

C.3 Additional results

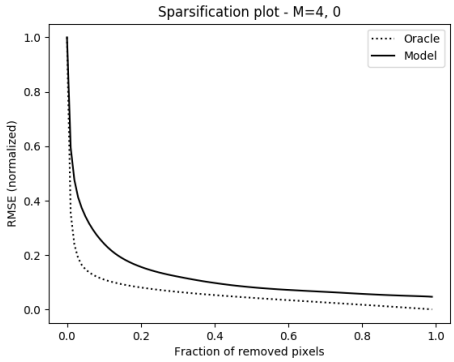
Here, we show sparsification plots, sparsification error curves and calibration plots. Examples of sparsification plots are found in Figure 19 for ensembling and Figure 20 for MC-dropout. Condensed sparsification error curves are found in Figure 21 for ensembling and Figure 22 for MC-dropout. Condensed calibration plots are found in Figure 23 for ensembling and Figure 24 for MC-dropout.



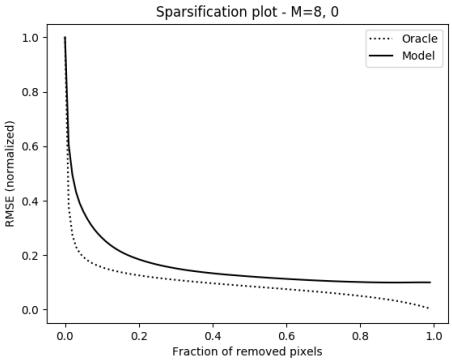
(a) $M = 1$.



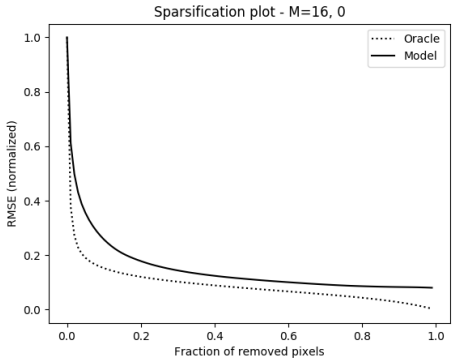
(b) $M = 2$.



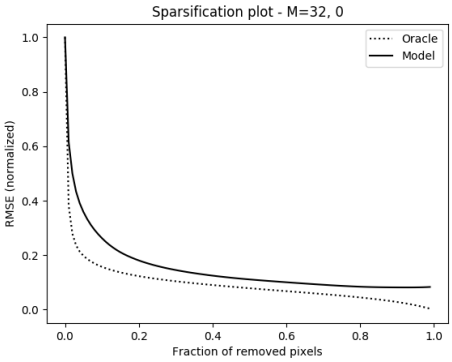
(c) $M = 4$.



(d) $M = 8$.

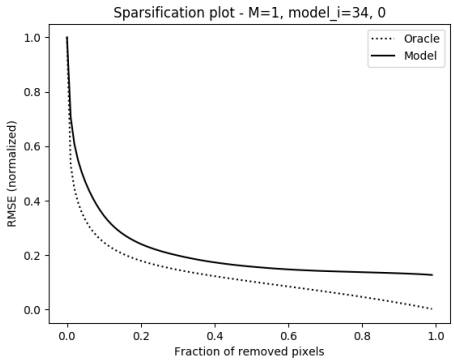


(e) $M = 16$.

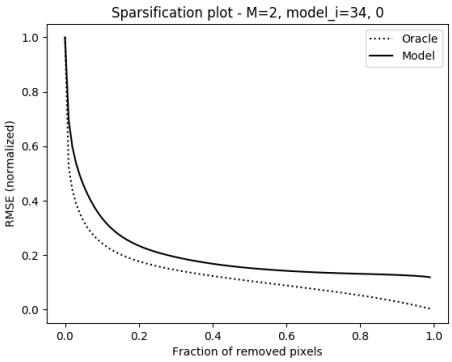


(f) $M = 32$.

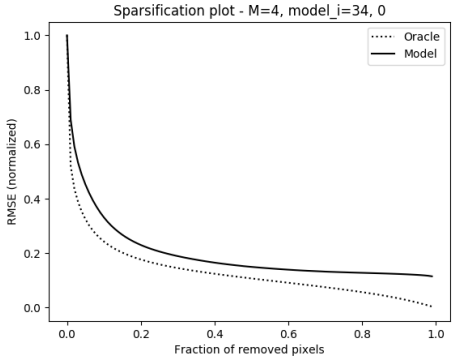
Figure 19: Results for ensembling on the KITTI depth completion validation dataset. Examples of sparsification plots.



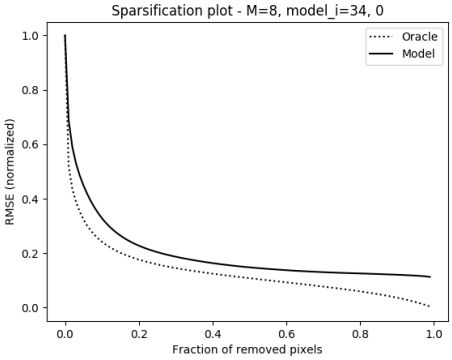
(a) $M = 1$.



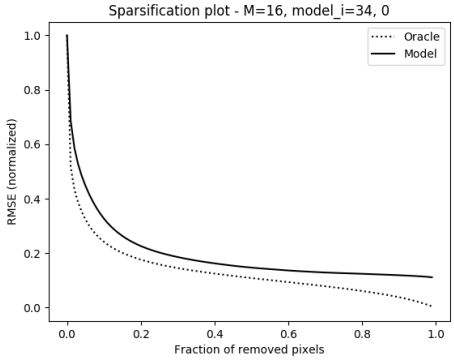
(b) $M = 2$.



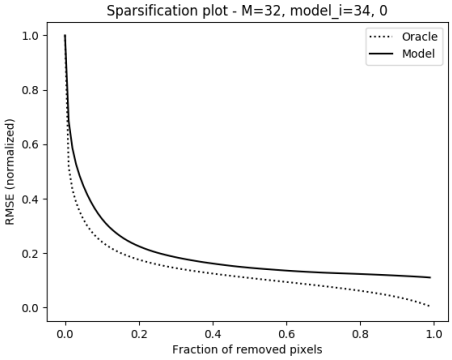
(c) $M = 4$.



(d) $M = 8$.

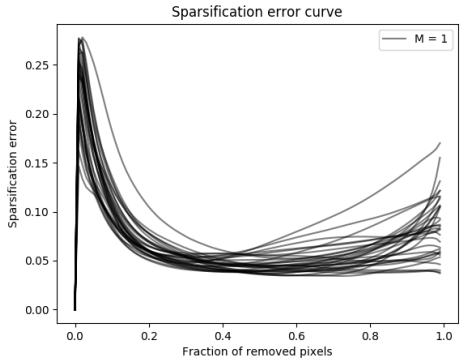


(e) $M = 16$.

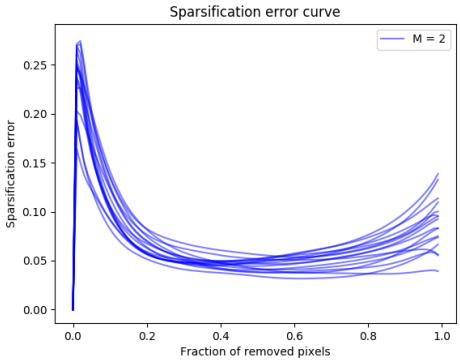


(f) $M = 32$.

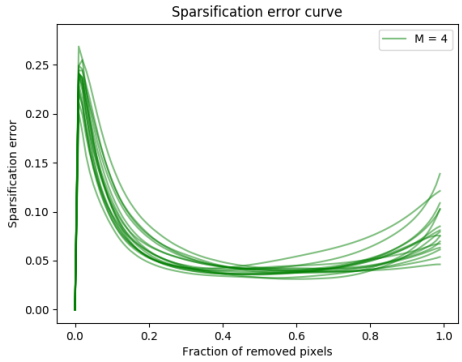
Figure 20: Results for MC-dropout on the KITTI depth completion validation dataset. Examples of sparsification plots.



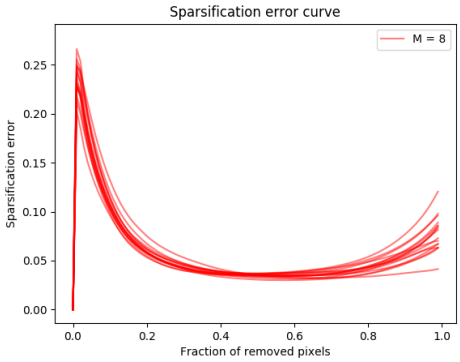
(a) $M = 1$.



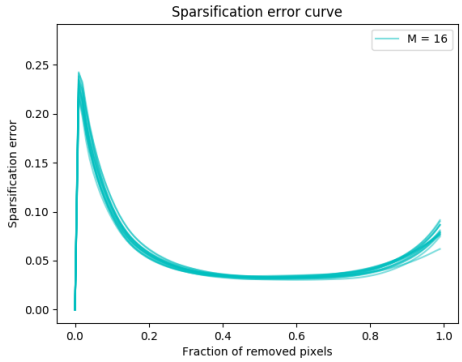
(b) $M = 2$.



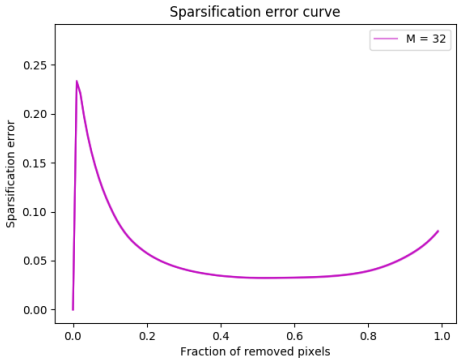
(c) $M = 4$.



(d) $M = 8$.

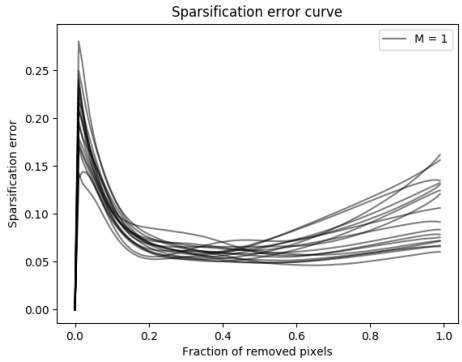


(e) $M = 16$.

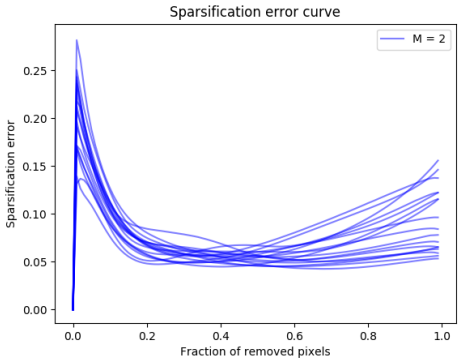


(f) $M = 32$.

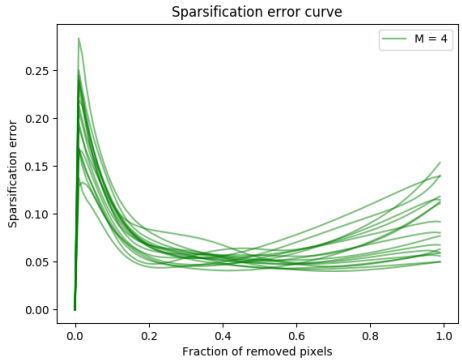
Figure 21: Results for ensembling on the KITTI depth completion validation dataset. Condensed sparsification error curves.



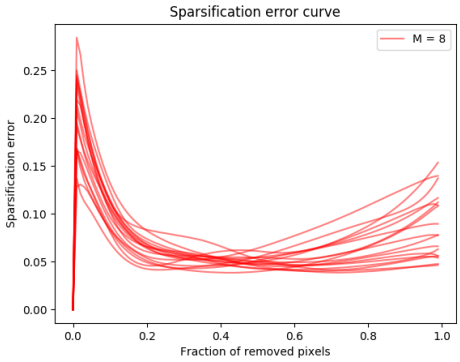
(a) $M = 1$.



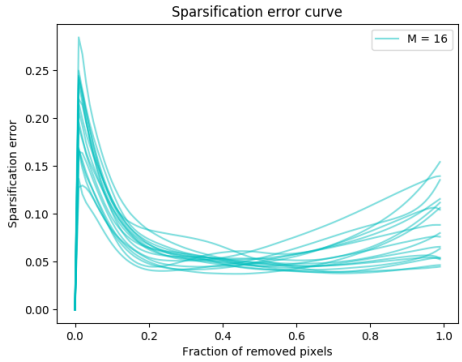
(b) $M = 2$.



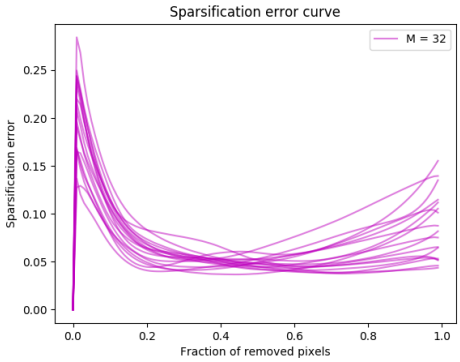
(c) $M = 4$.



(d) $M = 8$.

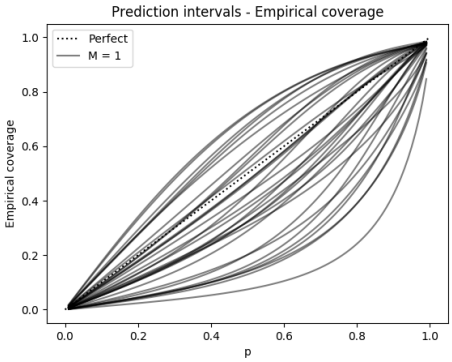


(e) $M = 16$.

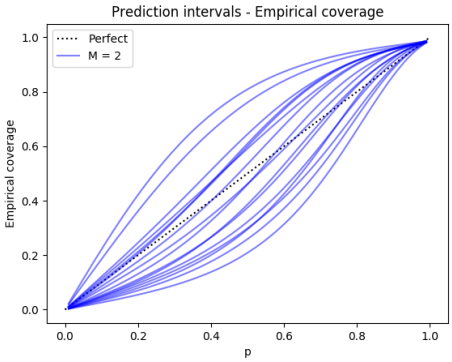


(f) $M = 32$.

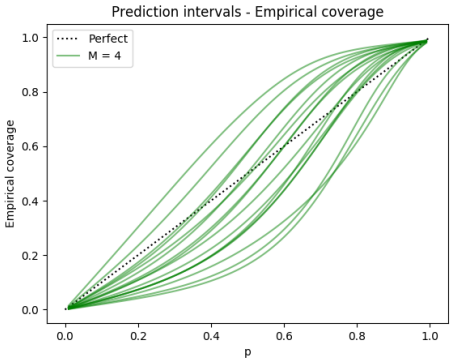
Figure 22: Results for MC-dropout on the KITTI depth completion validation dataset. Condensed sparsification error curves.



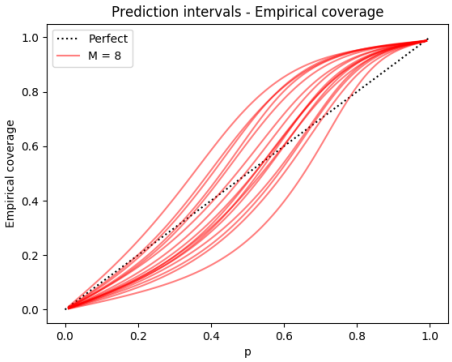
(a) $M = 1$.



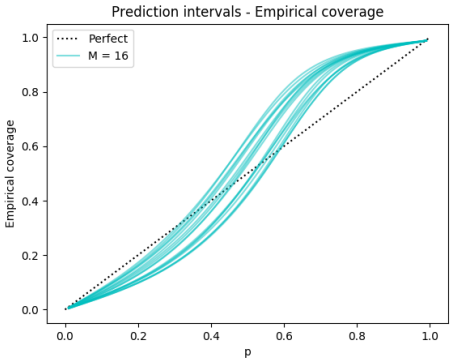
(b) $M = 2$.



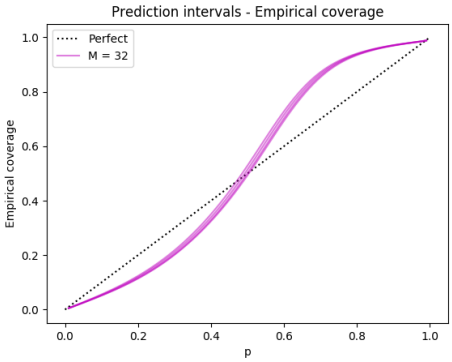
(c) $M = 4$.



(d) $M = 8$.

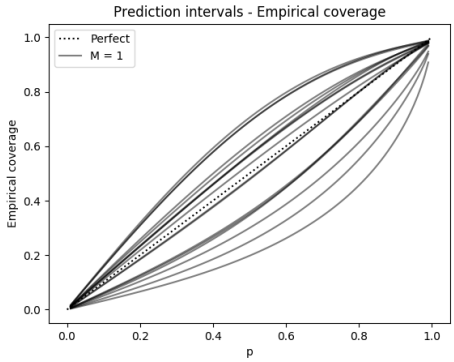


(e) $M = 16$.

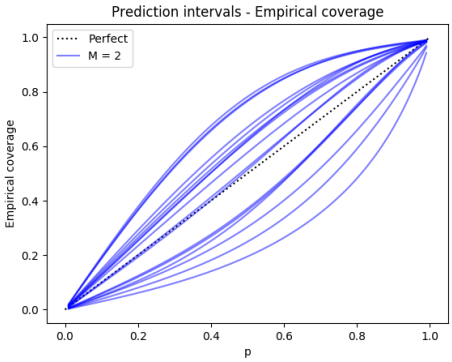


(f) $M = 32$.

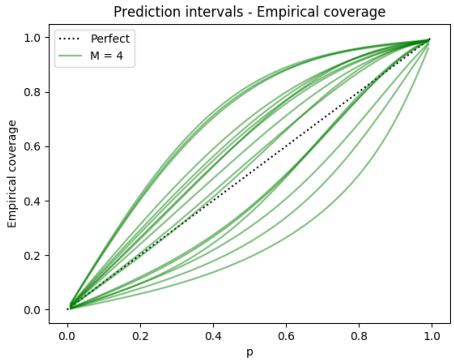
Figure 23: Results for ensembling on the KITTI depth completion validation dataset. Condensed calibration plots.



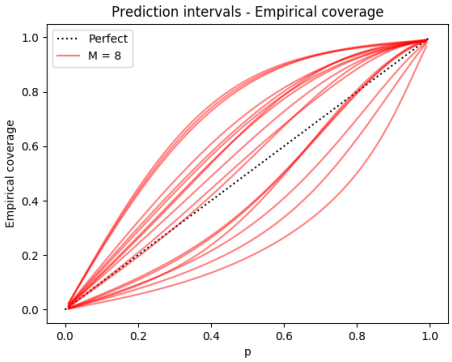
(a) $M = 1$.



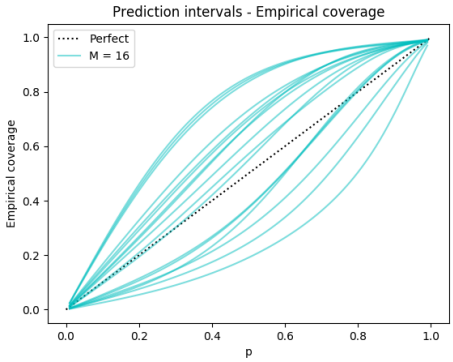
(b) $M = 2$.



(c) $M = 4$.



(d) $M = 8$.



(e) $M = 16$.

Figure 24: Results for MC-dropout on the KITTI depth completion validation dataset. Condensed calibration plots.

Appendix D Street-scene semantic segmentation

In this appendix, further details on the street-scene semantic segmentation experiments (Section 4.3) are provided.

D.1 Training details

For ensembling, we train all ensemble models for $40\,000$ steps with SGD + momentum (0.9), a batch size of 8 and weight decay of 0.0005. The learning rate α_t is decayed according to:

$$\alpha_t = \alpha_0(1 - \frac{t}{T})^{0.9}, \quad t = 1, 2, \ldots, T,$$

where $T = 40\,000$ and $\alpha_0 = 0.01$ (the initial learning rate). We train on randomly selected image crops of size 512×512. We choose a smaller crop size than Yuan and Wang [52] to enable an extensive evaluation with repeated experiments. The only other data augmentation used is random flipping along the vertical axis and random scaling in the range $[0.5, 1.5]$. The ResNet101 backbone is initialized with weights[1] from a model pretrained on the ImageNet dataset, all other model parameters are randomly initialized using the default initializer in PyTorch. Models are trained on two NVIDIA TITAN Xp GPUs with 12GB of RAM each. For MC-dropout, models are instead trained for $60\,000$ steps.

D.2 Description of results

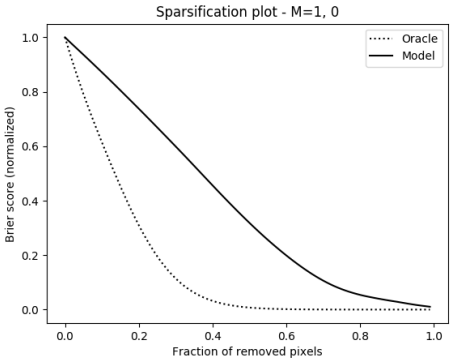
The results in Figure 6 (Section 4.3) were obtained in the following way:

- **Ensembling:** 26 models were trained using the same training procedure, the mean and standard deviation was computed based on 8 sets of randomly drawn models for $M \in \{1, 2, 4, 8, 16\}$. The same set could not be drawn more than once.
- **MC-dropout:** 8 models were trained using the same training procedure, based on which the mean and standard deviation was computed.

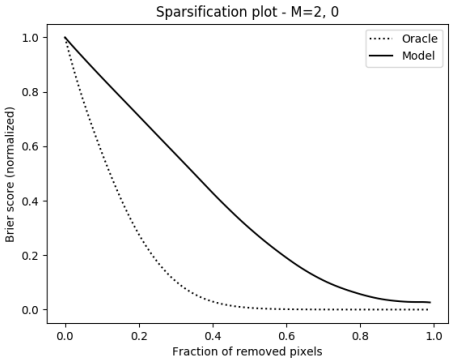
D.3 Additional results

Here, we show sparsification plots, sparsification error curves and reliability diagrams. Examples of sparsification plots are found in Figure 25 for ensembling and Figure 26 for MC-dropout. Condensed sparsification error curves are found in Figure 27 for ensembling and Figure 28 for MC-dropout. Examples of reliability diagrams with histograms are found in Figure 29 for ensembling and Figure 30 for MC-dropout. Condensed reliability diagrams are found in Figure 31 for ensembling and Figure 32 for MC-dropout.

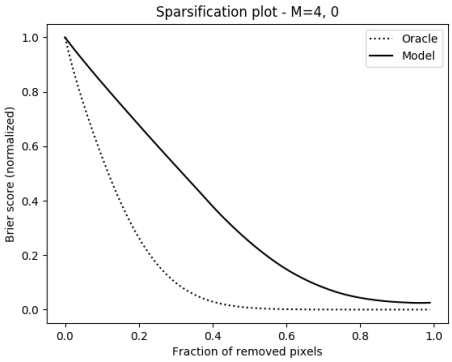
[1] `http://sceneparsing.csail.mit.edu/model/pretrained_resnet/resnet101-imagenet.pth`.



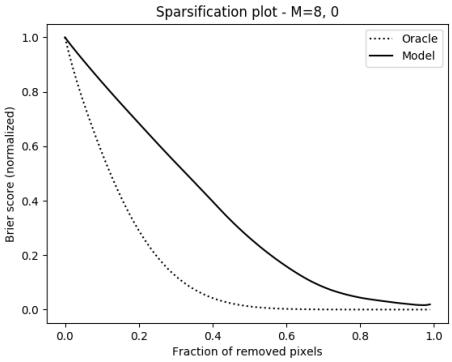
(a) $M = 1$.



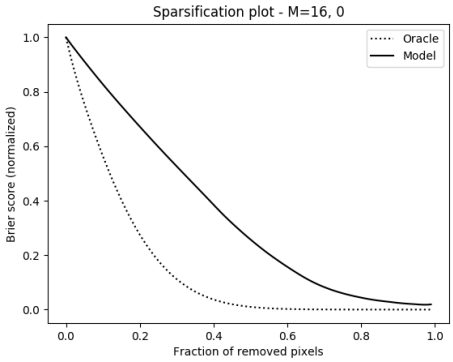
(b) $M = 2$.



(c) $M = 4$.

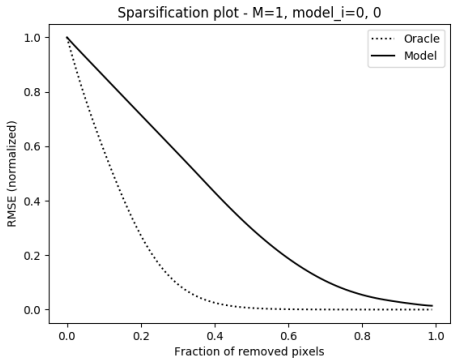


(d) $M = 8$.

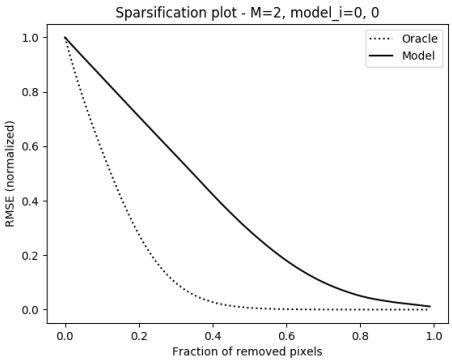


(e) $M = 16$.

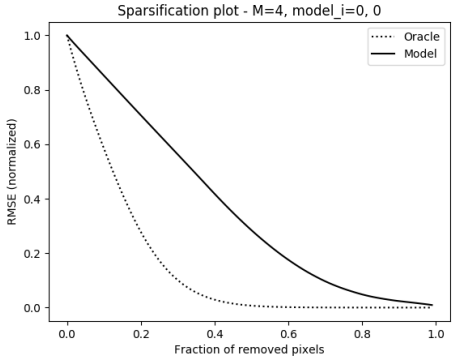
Figure 25: Results for ensembling on the Cityscapes validation dataset. Examples of sparsification plots.



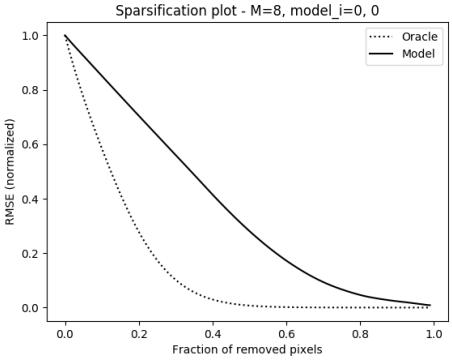
(a) $M = 1$.



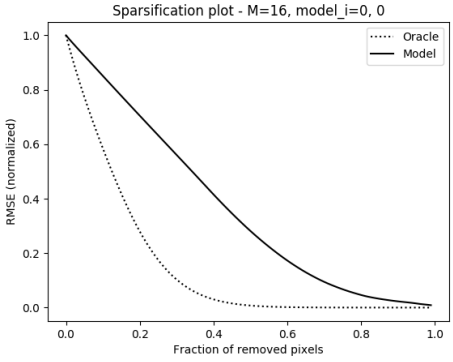
(b) $M = 2$.



(c) $M = 4$.

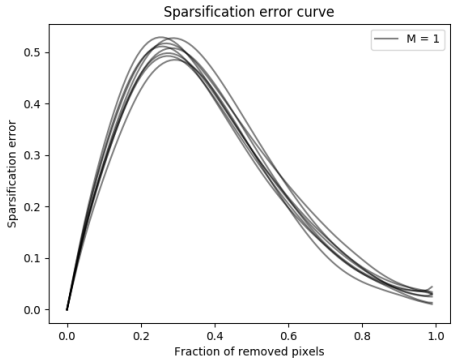


(d) $M = 8$.

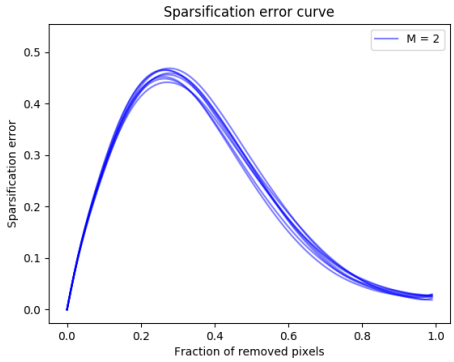


(e) $M = 16$.

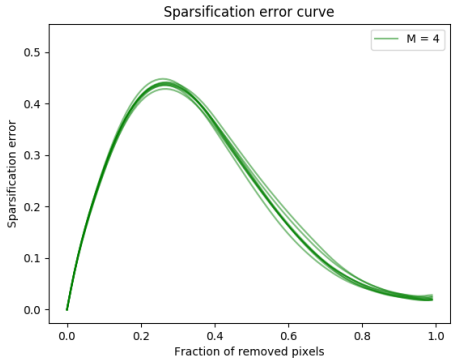
Figure 26: Results for MC-dropout on the Cityscapes validation dataset. Examples of sparsification plots.



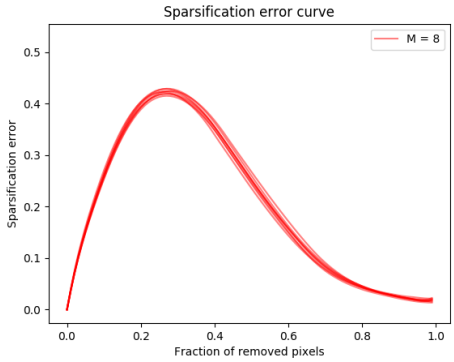
(a) $M = 1$.



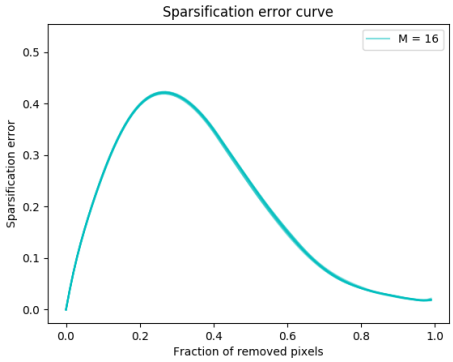
(b) $M = 2$.



(c) $M = 4$.

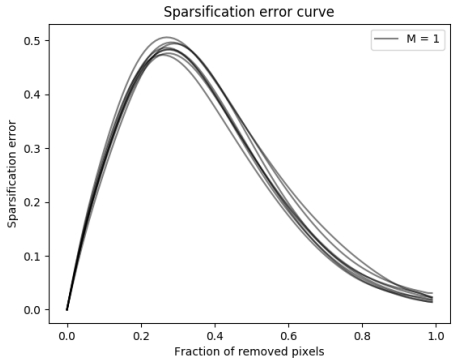


(d) $M = 8$.

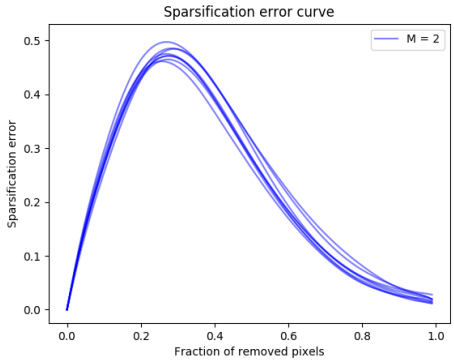


(e) $M = 16$.

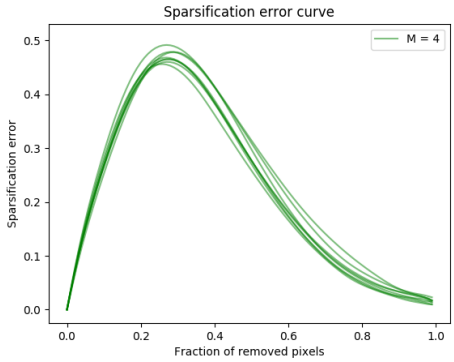
Figure 27: Results for ensembling on the Cityscapes validation dataset. Condensed sparsification error curves.



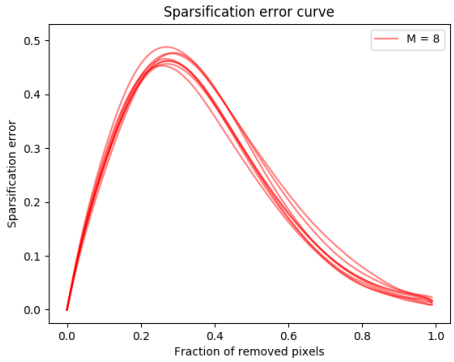
(a) $M = 1$.



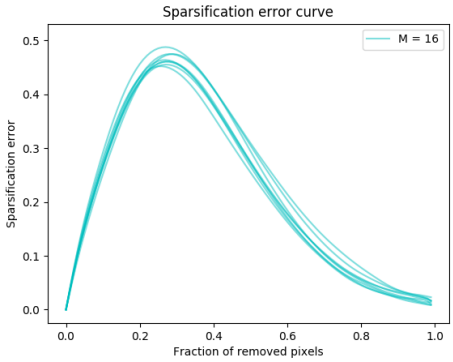
(b) $M = 2$.



(c) $M = 4$.

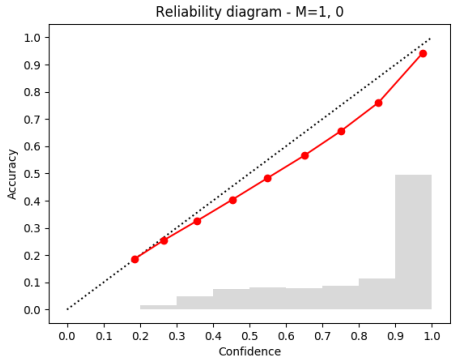


(d) $M = 8$.

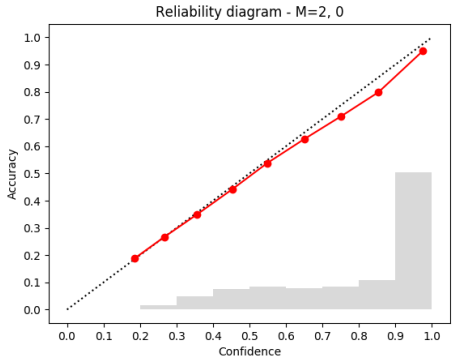


(e) $M = 16$.

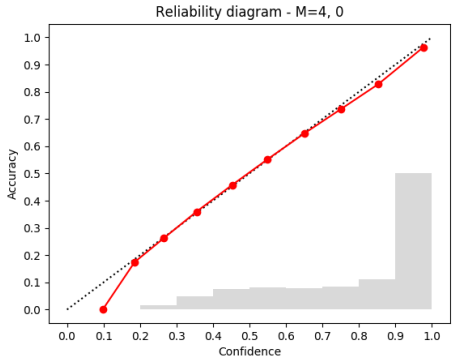
Figure 28: Results for MC-dropout on the Cityscapes validation dataset. Condensed sparsification error curves.



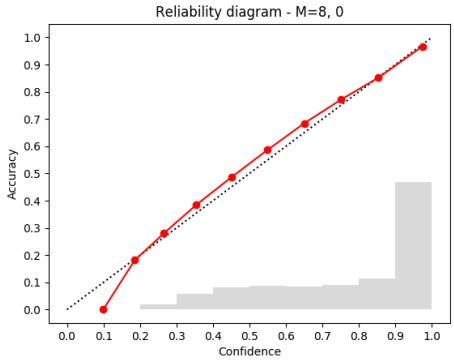
(a) $M = 1$.



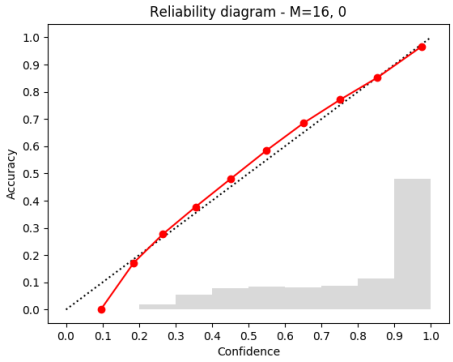
(b) $M = 2$.



(c) $M = 4$.

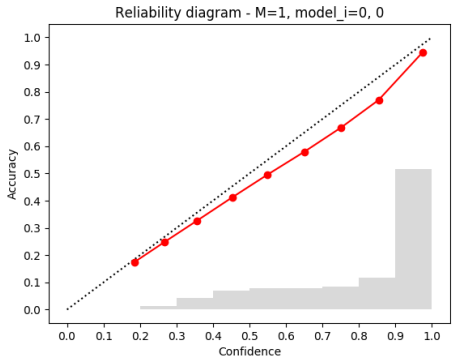


(d) $M = 8$.

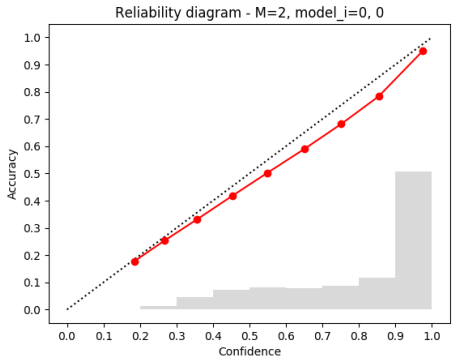


(e) $M = 16$.

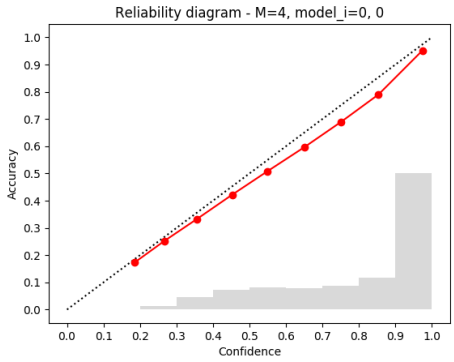
Figure 29: Results for ensembling on the Cityscapes validation dataset. Examples of reliability diagrams with histograms.



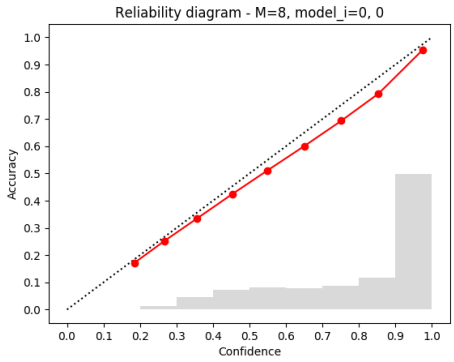
(a) $M = 1$.



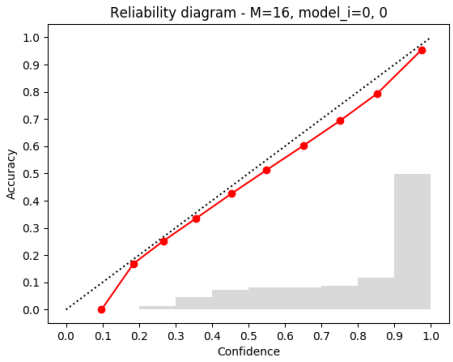
(b) $M = 2$.



(c) $M = 4$.

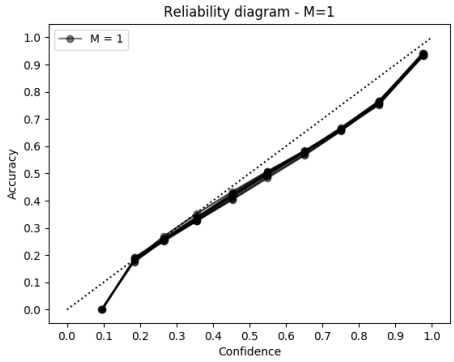


(d) $M = 8$.

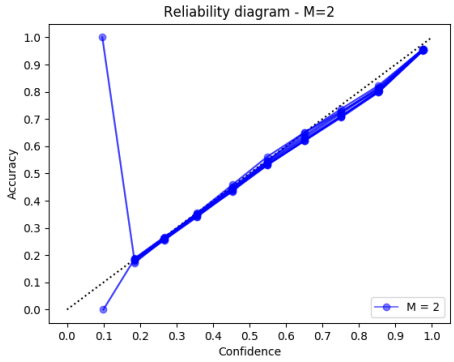


(e) $M = 16$.

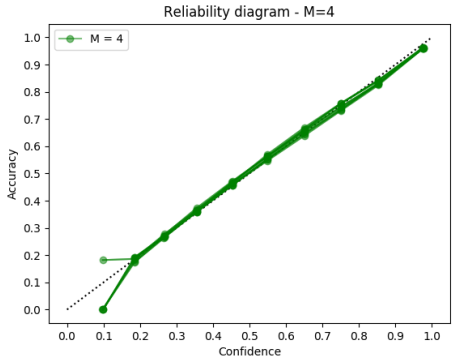
Figure 30: Results for MC-dropout on the Cityscapes validation dataset. Examples of reliability diagrams with histograms.



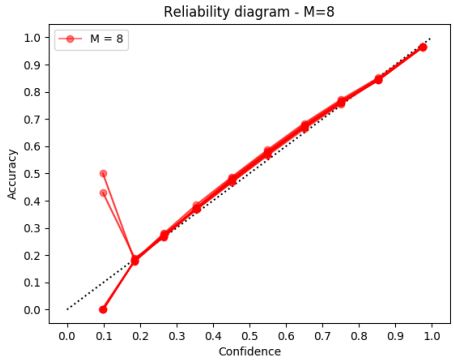
(a) $M = 1$.



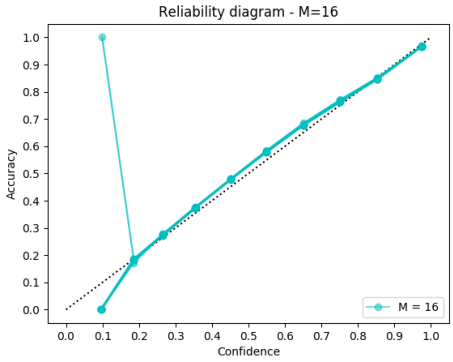
(b) $M = 2$.



(c) $M = 4$.

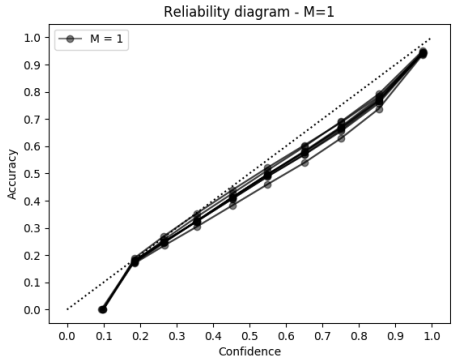


(d) $M = 8$.

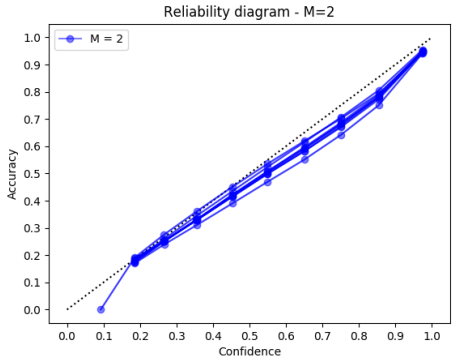


(e) $M = 16$.

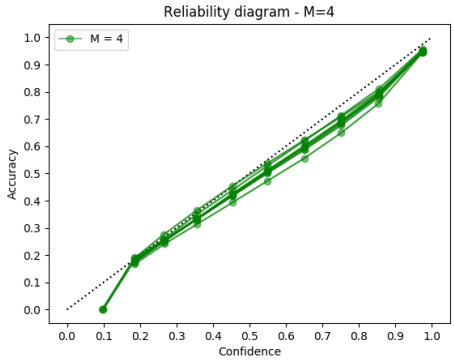
Figure 31: Results for ensembling on the Cityscapes validation dataset. Condensed reliability diagrams.



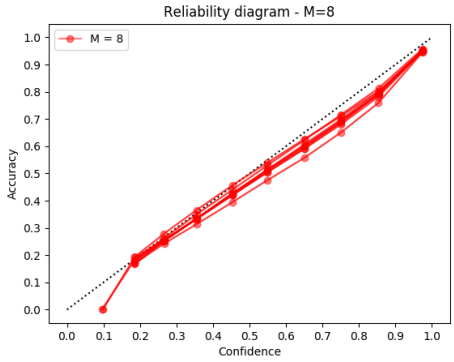
(a) $M = 1$.



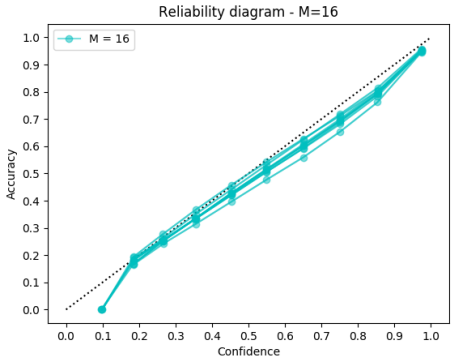
(b) $M = 2$.



(c) $M = 4$.



(d) $M = 8$.



(e) $M = 16$.

Figure 32: Results for MC-dropout on the Cityscapes validation dataset. Condensed reliability diagrams.