Supplementary Material

This is the supplementary material for ***Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision***. It consists of Appendix A-D.

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:

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

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

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 5a (regression) shows $D_{\mathrm{KL}}(p \parallel p_{\mathrm{HMC}})$ computed on $[-7, 7]$. All training data was given in $[-3, 3]$.

Figure 5b (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 $y_i = [\, y_{i,1} \ldots y_{i,C}\,]^\mathsf{T}$ (one-hot encoded) and $\hat{s}(x_i) = [\, \hat{s}(x_i)_1 \ldots \hat{s}(x_i)_C\,]^\mathsf{T}$ 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:

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

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:

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

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 5a, 5b 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 10: Illustrative toy problems, quantitative results. SGD is used for ensembling and MC-dropout instead of Adam.

(a) Regression (b) Classification

Figure 11: Illustrative toy problems, quantitative results. SGD with momentum is used for ensembling and MC-dropout instead of Adam.

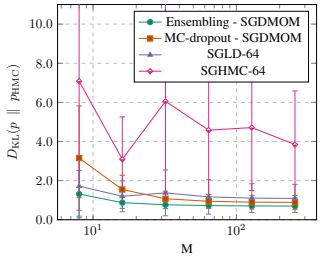


Figure 12: Illustrative toy regression problem, quantitative results. SGD with momentum is used for ensembling and MC-dropout instead of Adam. Less training for SGLD and SGHMC.

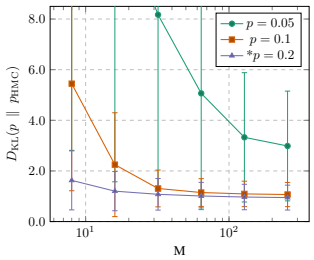
B.4. Additional Results

Figure 10 and Figure 11 show the same comparison as Figure 5a, 5b, 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 12 that they are consistently outperformed by ensembling.

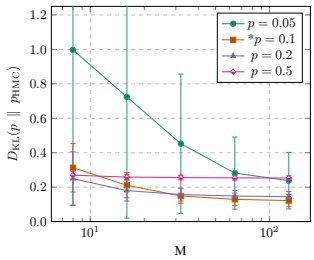
For MC-dropout using Adam, we also varied the drop probability p and chose the best performing variant. These results are found in Figure 13, in which * marks the chosen variant.

B.5. Qualitative Results

Here, we show visualizations of predictive distributions obtained by the different methods. Figure 14, 18 for ensembling, Figure 15, 19 for MC-dropout, Figure 16, 20 for SGLD, and Figure 17, 21 for SGHMC.



(a) Regression



(b) Classification

Figure 13: Illustrative toy problems, quantitative results. MC-dropout using Adam.

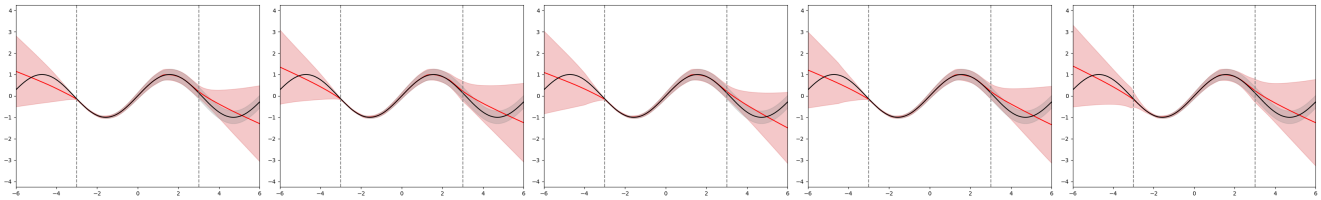


Figure 14: Toy regression problem, ensembling, $M = 64$. Examples of predictive distributions.

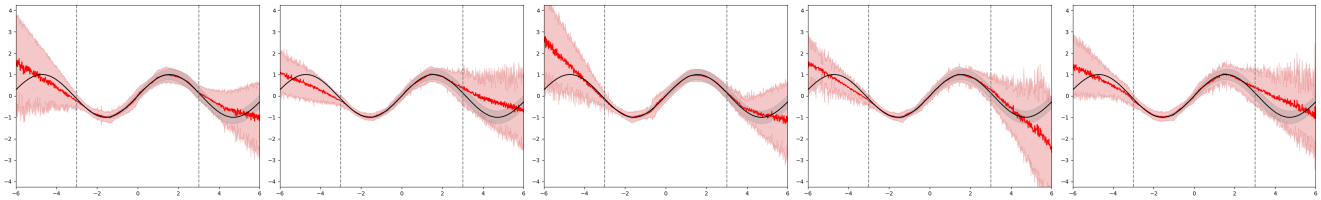


Figure 15: Toy regression problem, MC-dropout, $M = 64$. Examples of predictive distributions.

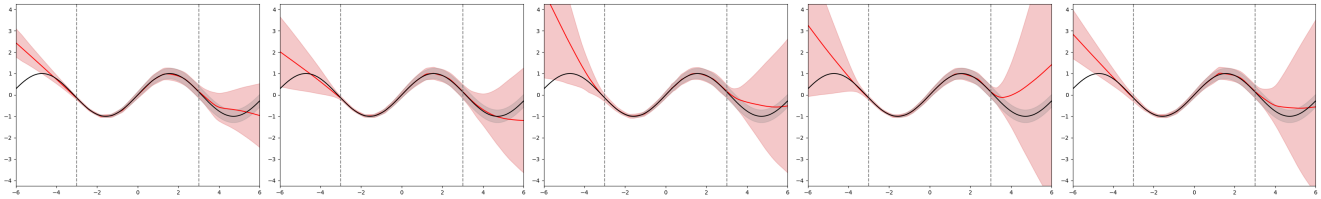


Figure 16: Toy regression problem, SGLD, $M = 64$. Examples of predictive distributions.

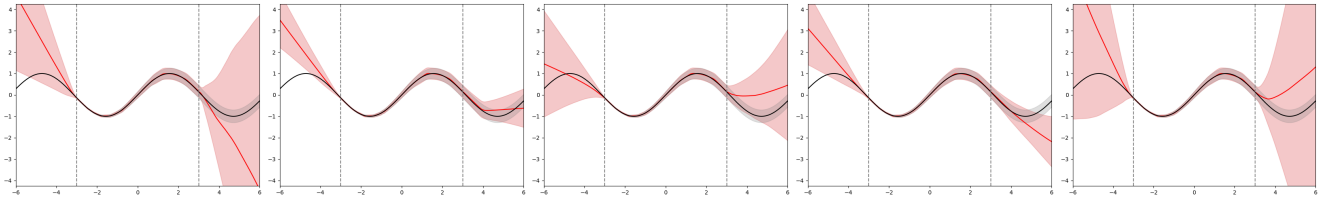


Figure 17: Toy regression problem, SGHMC, $M = 64$. Examples of predictive distributions.

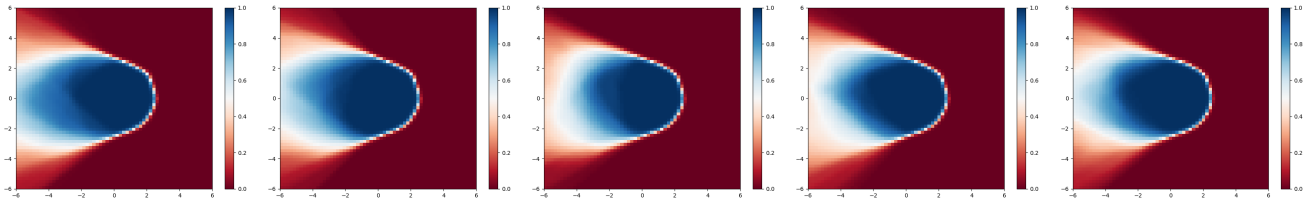


Figure 18: Toy classification problem, ensembling, $M = 64$. Examples of predictive distributions.

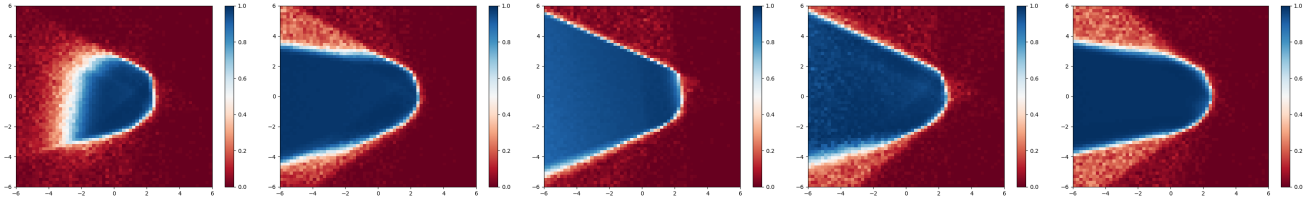


Figure 19: Toy classification problem, MC-dropout, $M = 64$. Examples of predictive distributions.

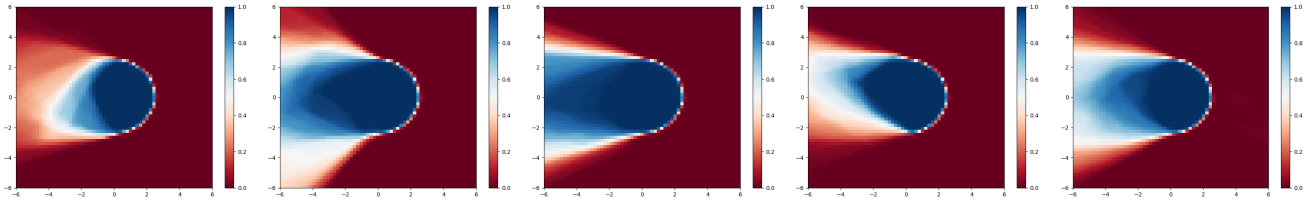


Figure 20: Toy classification problem, SGLD, $M = 64$. Examples of predictive distributions.

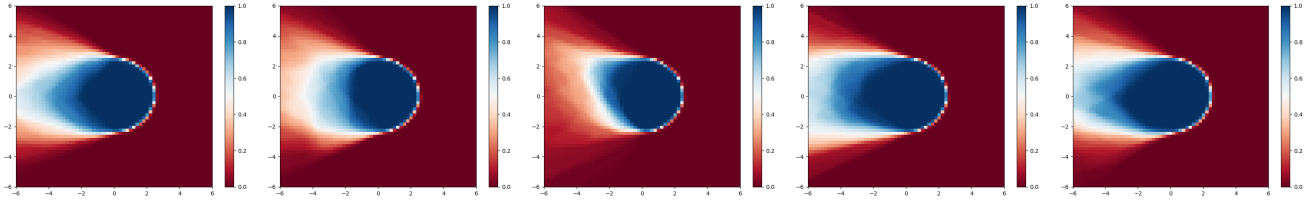


Figure 21: Toy classification problem, SGHMC, $M = 64$. Examples of predictive distributions.

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. [32] 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. [32] 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.

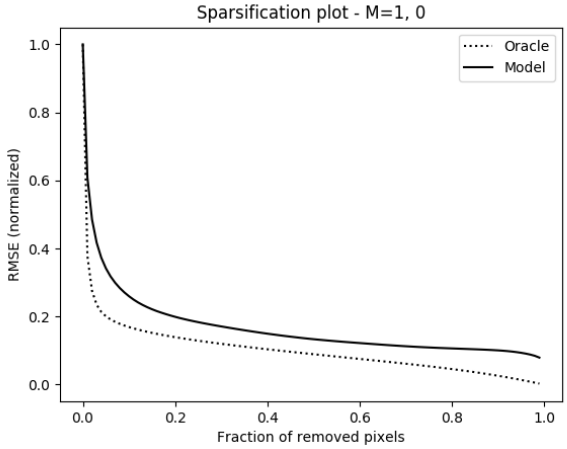
C.2. Description of Results

The results in Figure 6 (Section 4.2) were obtained in the following way:

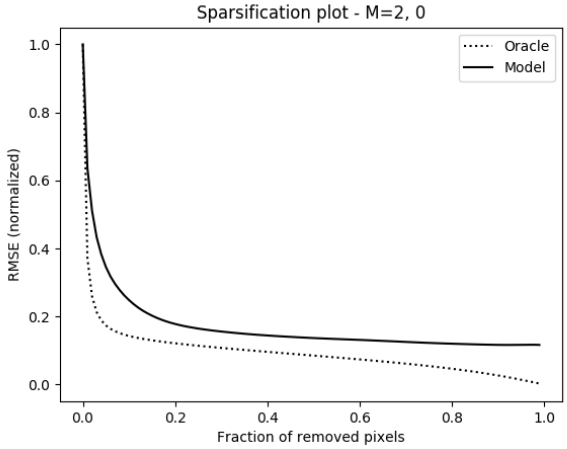
- **Ensembling:** 33 models were trained using the same training procedure, the mean and standard deviation was computed based on 32 ($M = 1$), 16 ($M = 2, 4, 8, 16$) or 4 ($M = 32$) sets of randomly drawn models. The same set could not be drawn more than once.
- **MC-dropout:** 16 models were trained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

C.3. Additional Results

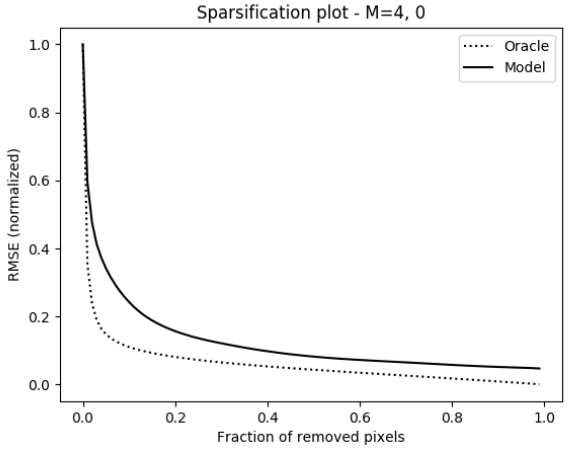
Here, we show sparsification plots, sparsification error curves and calibration plots. Examples of sparsification plots are found in Figure 22 for ensembling and Figure 23 for MC-dropout. Condensed sparsification error curves are found in Figure 24 for ensembling and Figure 25 for MC-dropout. Condensed calibration plots are found in Figure 26 for ensembling and Figure 27 for MC-dropout.



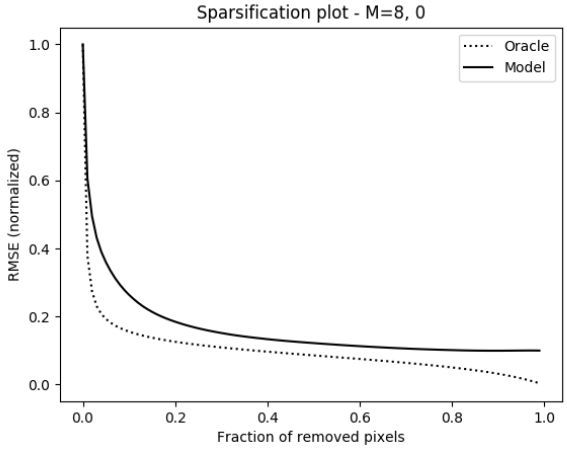
(a) $M = 1$.



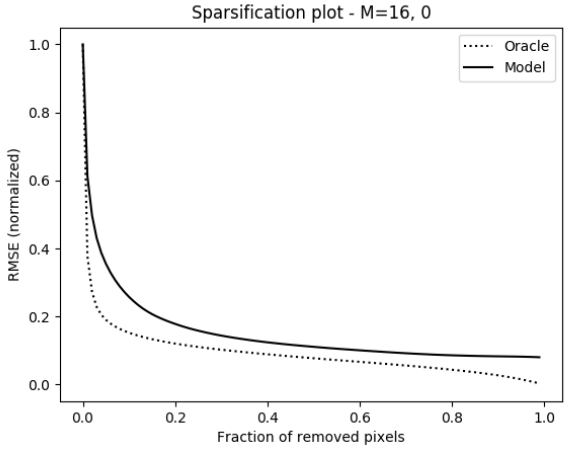
(b) $M = 2$.



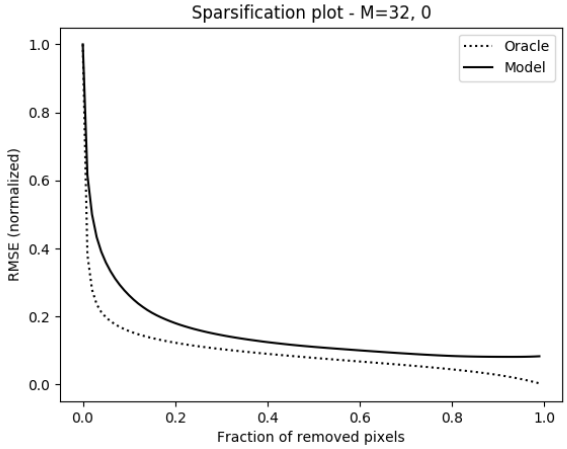
(c) $M = 4$.



(d) $M = 8$.

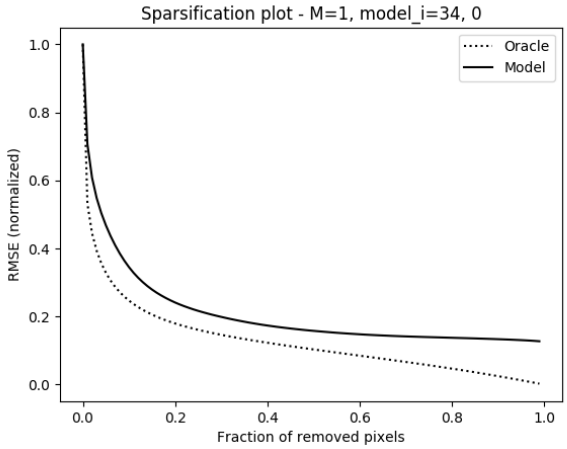


(e) $M = 16$.

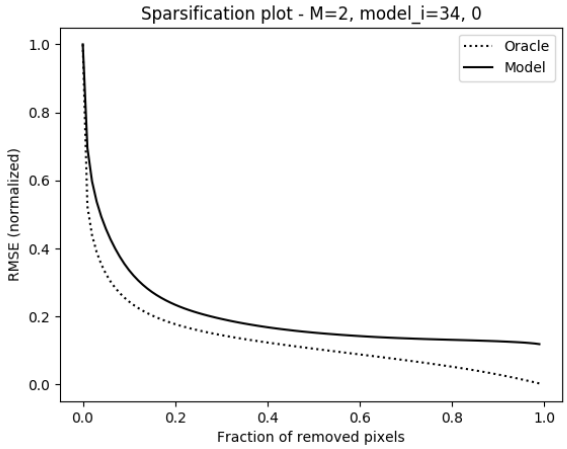


(f) $M = 32$.

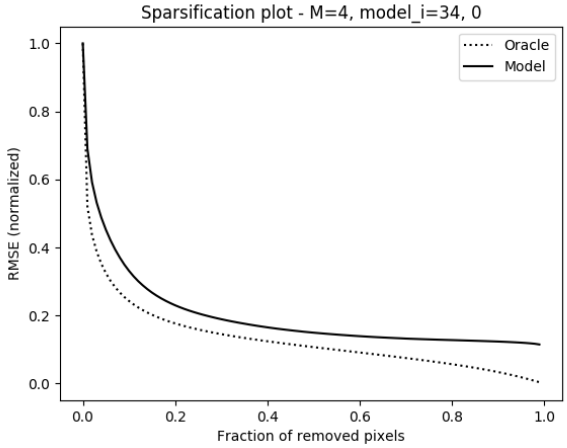
Figure 22: Results for ensembling on the KITTI depth completion validation dataset. Examples of sparsification plots.



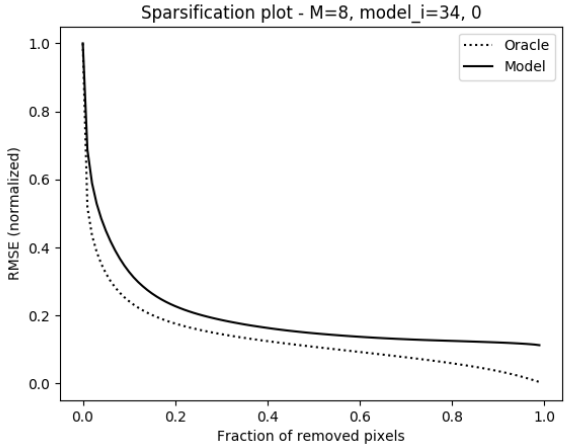
(a) $M = 1$.



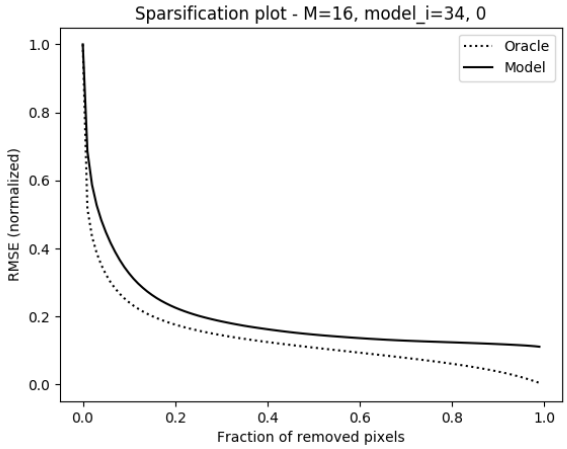
(b) $M = 2$.



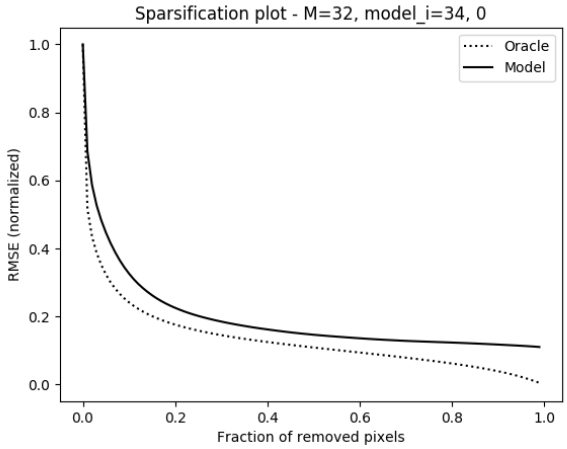
(c) $M = 4$.



(d) $M = 8$.

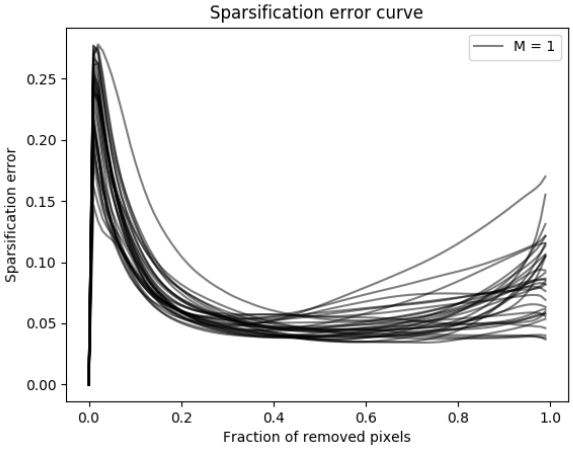


(e) $M = 16$.

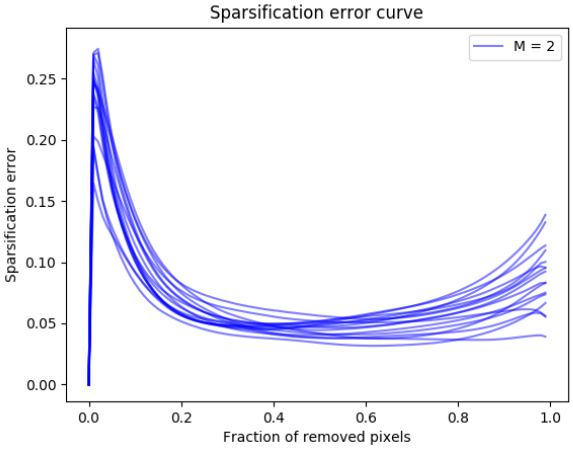


(f) $M = 32$.

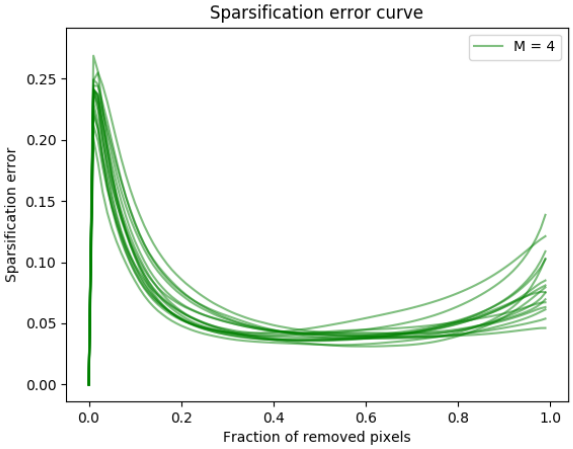
Figure 23: Results for MC-dropout on the KITTI depth completion validation dataset. Examples of sparsification plots.



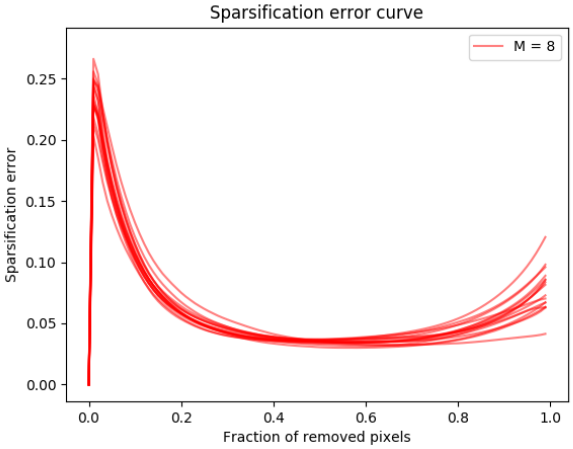
(a) $M = 1$.



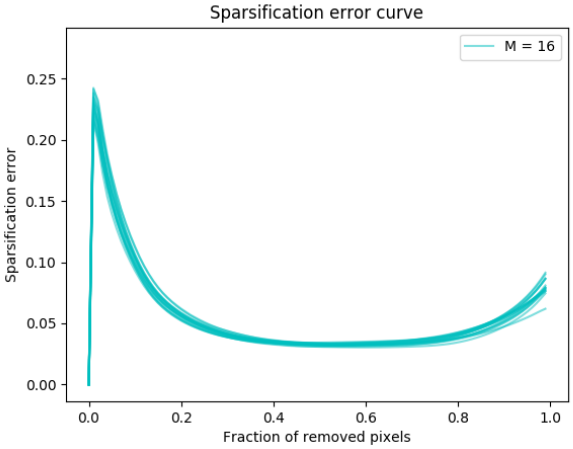
(b) $M = 2$.



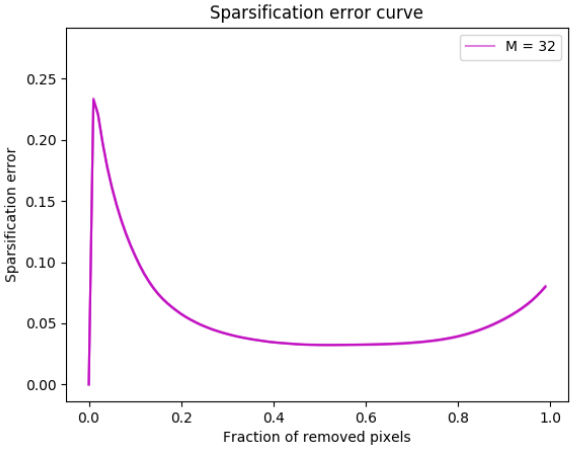
(c) $M = 4$.



(d) $M = 8$.

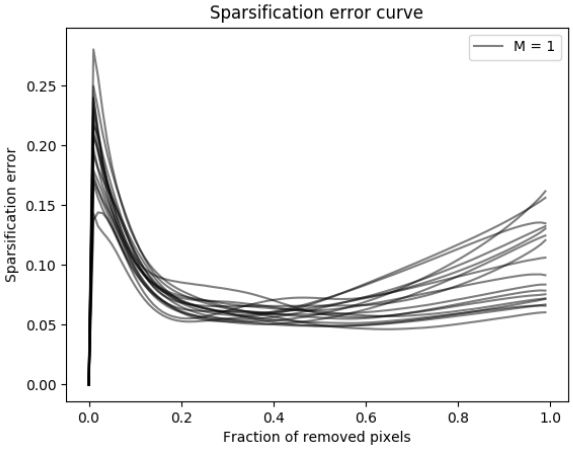


(e) $M = 16$.

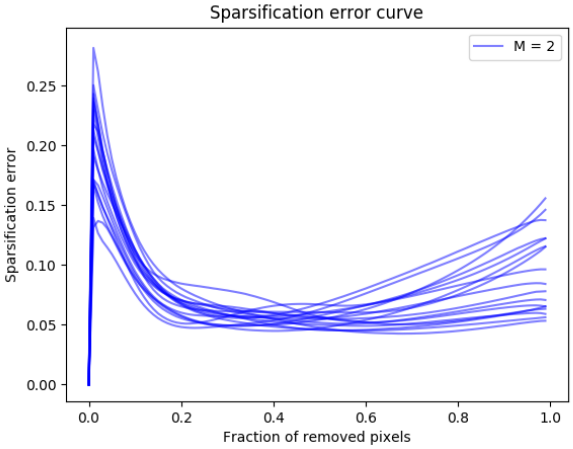


(f) $M = 32$.

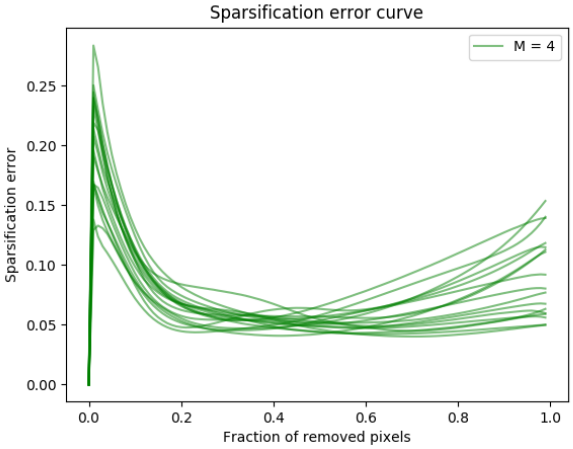
Figure 24: Results for ensembling on the KITTI depth completion validation dataset. Condensed sparsification error curves.



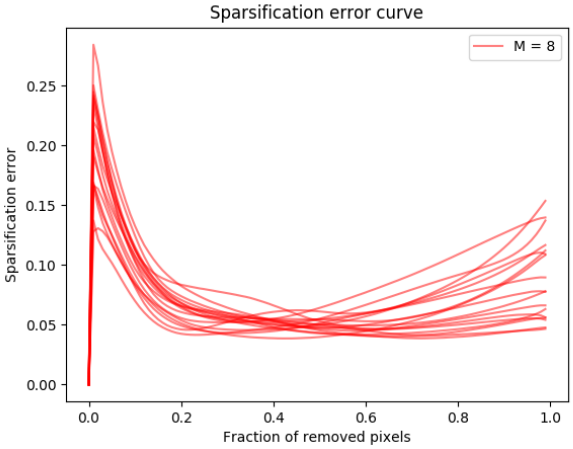
(a) $M = 1$.



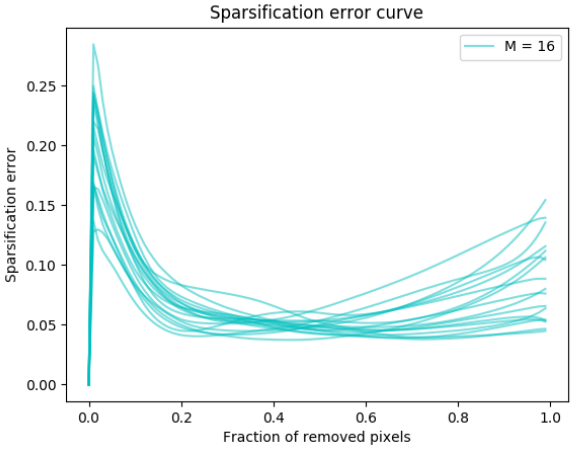
(b) $M = 2$.



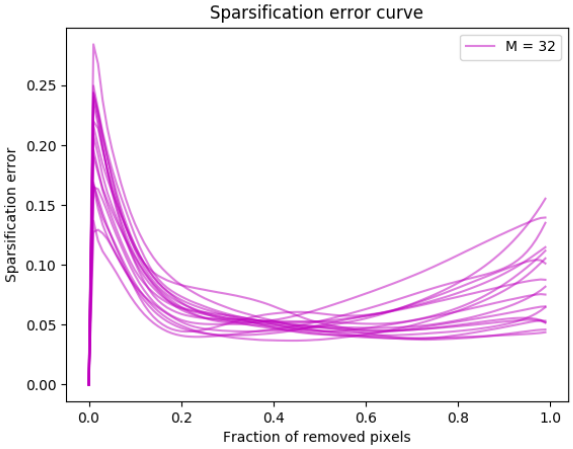
(c) $M = 4$.



(d) $M = 8$.

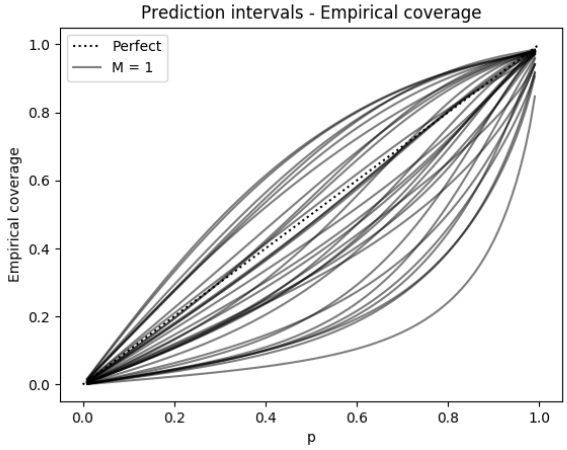


(e) $M = 16$.

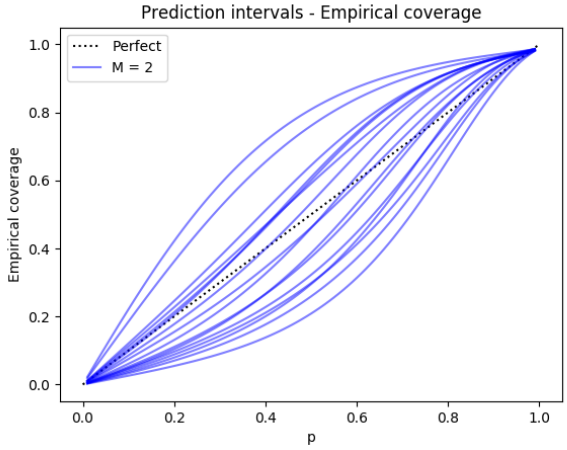


(f) $M = 32$.

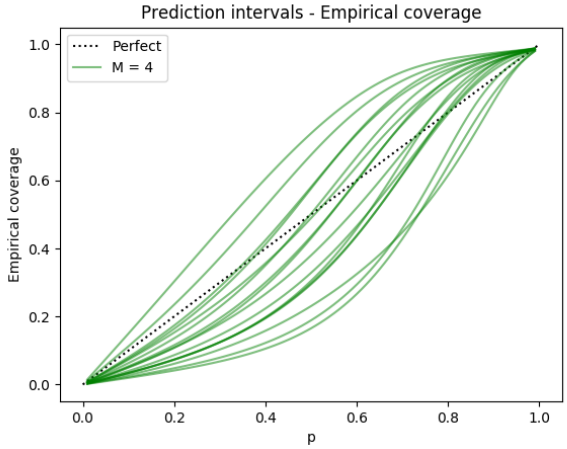
Figure 25: Results for MC-dropout on the KITTI depth completion validation dataset. Condensed sparsification error curves.



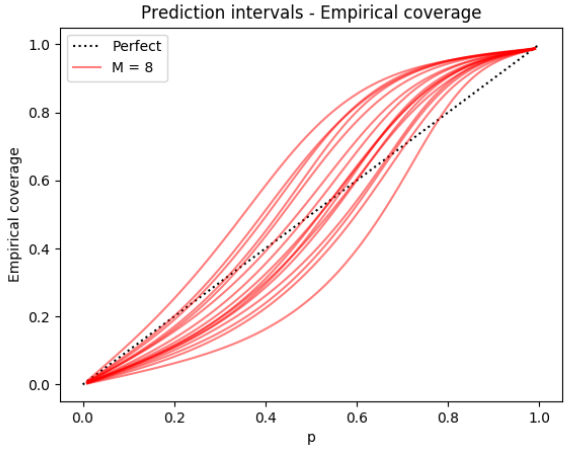
(a) $M = 1$.



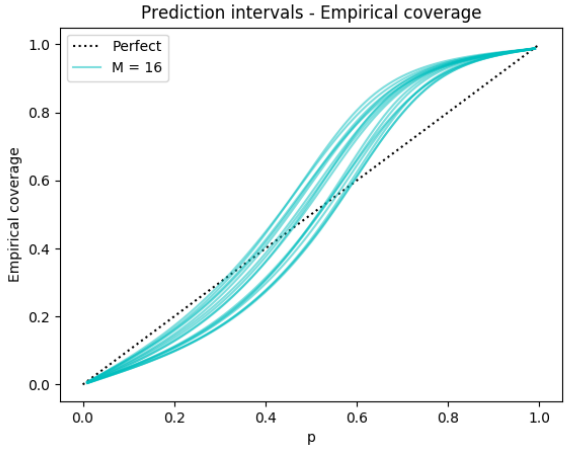
(b) $M = 2$.



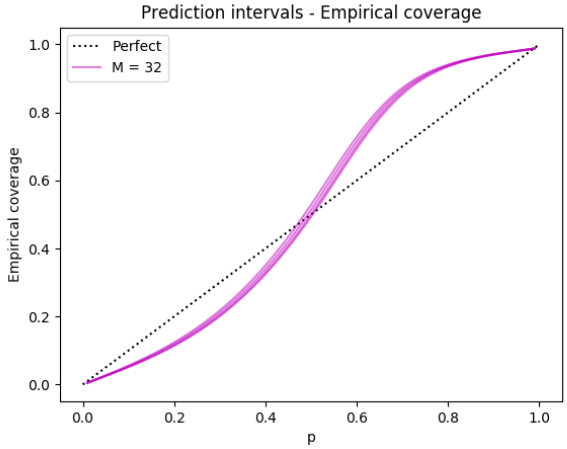
(c) $M = 4$.



(d) $M = 8$.

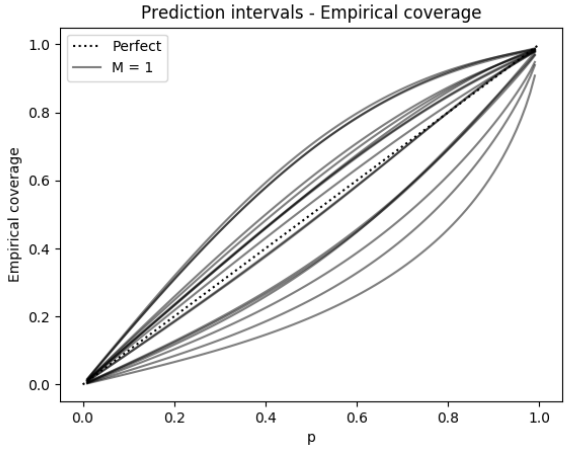


(e) $M = 16$.

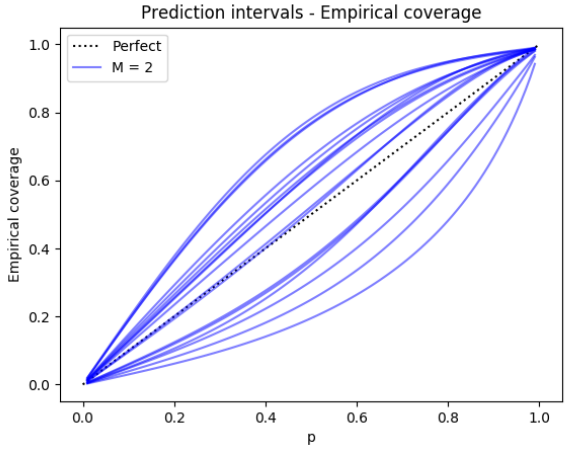


(f) $M = 32$.

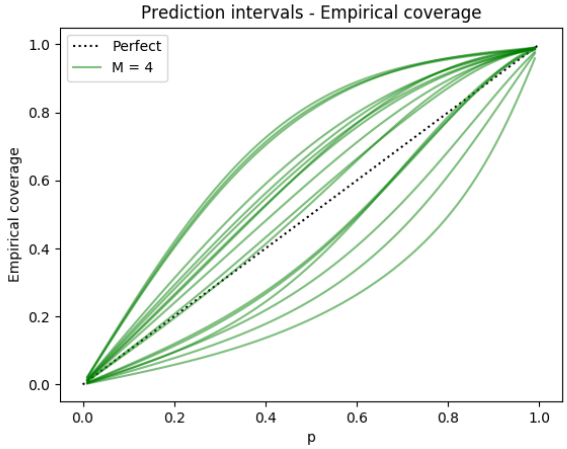
Figure 26: Results for ensembling on the KITTI depth completion validation dataset. Condensed calibration plots.



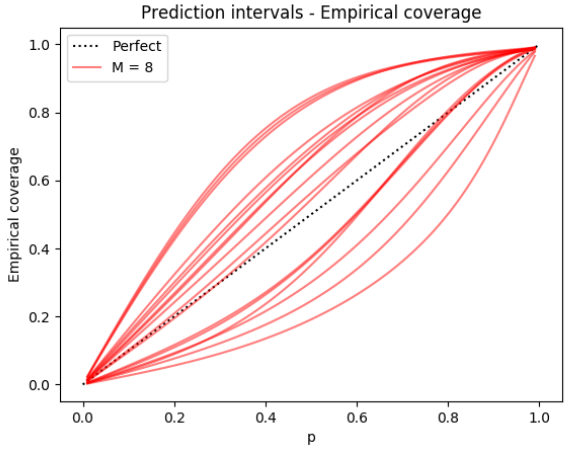
(a) $M = 1$.



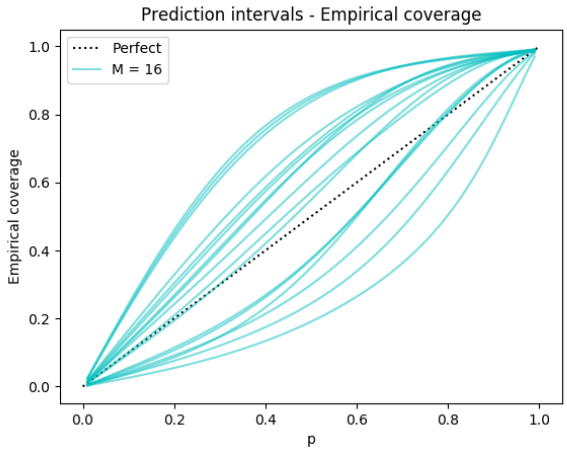
(b) $M = 2$.



(c) $M = 4$.



(d) $M = 8$.



(e) $M = 16$.

Figure 27: 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 [51] 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

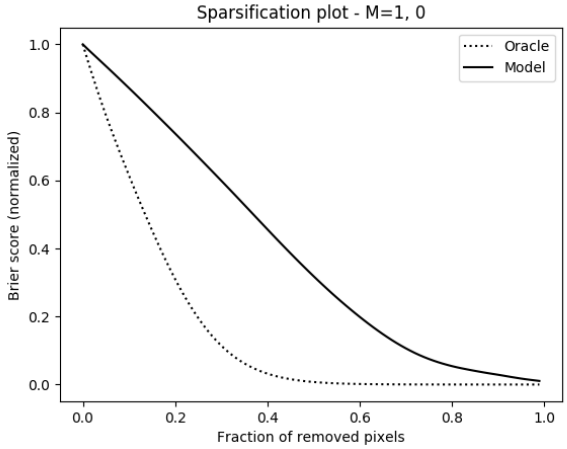
The results in Figure 8 (Section 4.3) were obtained in the following way:

- **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.

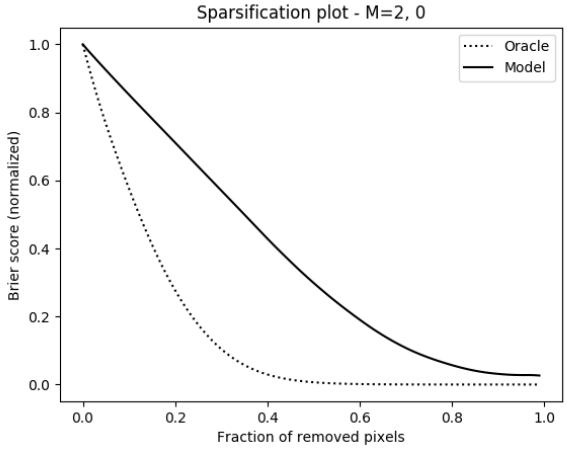
D.3. Additional Results

Here, we show sparsification plots, sparsification error curves and reliability diagrams. Examples of sparsification plots are found in Figure 28 for ensembling and Figure 29 for MC-dropout. Condensed sparsification error curves are found in Figure 30 for ensembling and Figure 31 for MC-dropout. Examples of reliability diagrams with histograms are found in Figure 32 for ensembling and Figure 33 for MC-dropout. Condensed reliability diagrams are found in Figure 34 for ensembling and Figure 35 for MC-dropout.

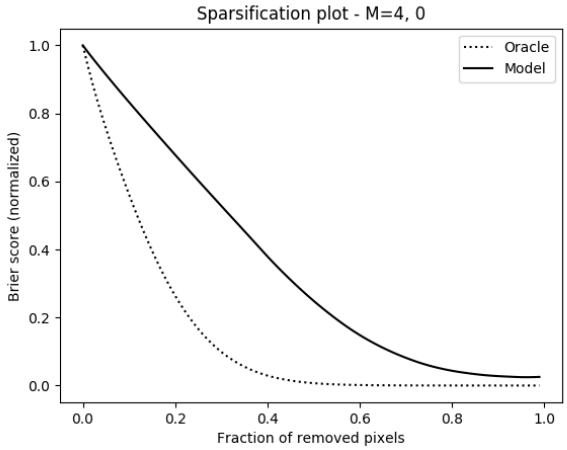
[1] http://sceneparsing.csail.mit.edu/model/pretrained_resnet/resnet101-imagenet.pth.



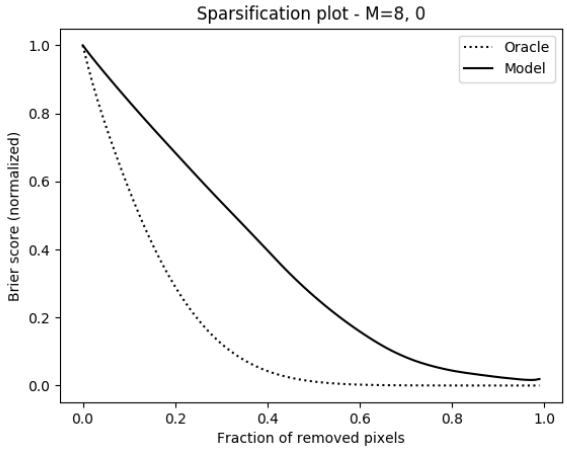
(a) $M = 1$.



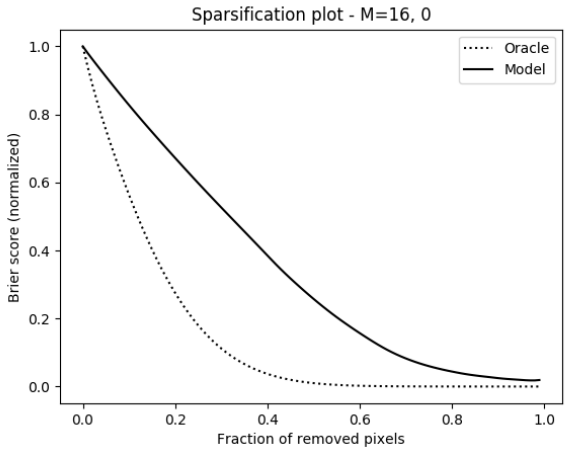
(b) $M = 2$.



(c) $M = 4$.

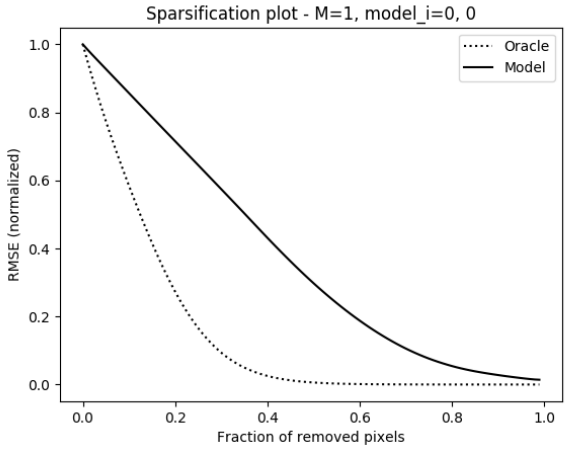


(d) $M = 8$.

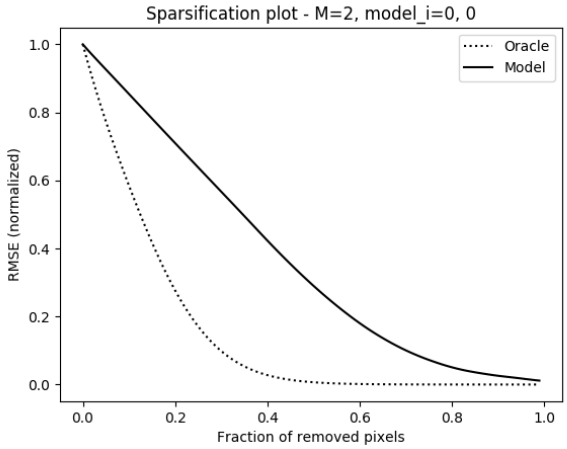


(e) $M = 16$.

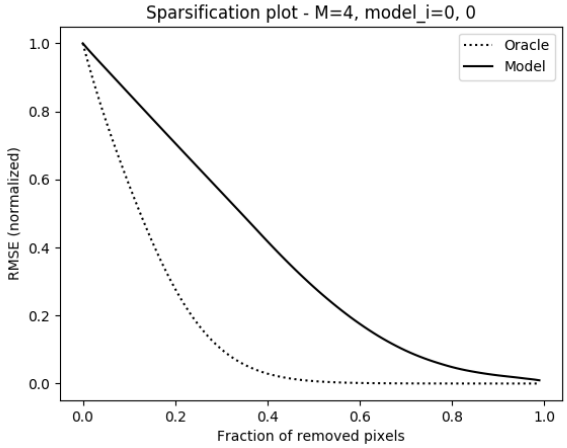
Figure 28: Results for ensembling on the Cityscapes validation dataset. Examples of sparsification plots.



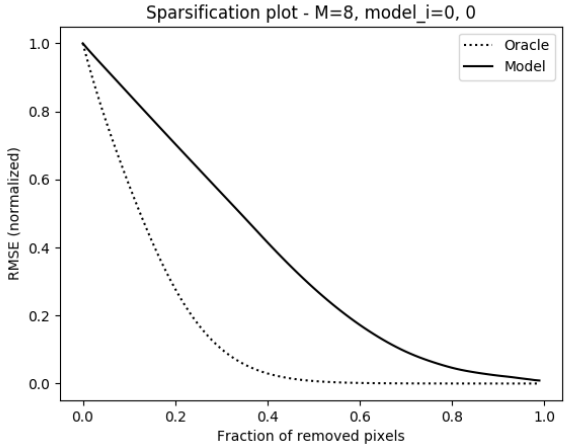
(a) $M = 1$.



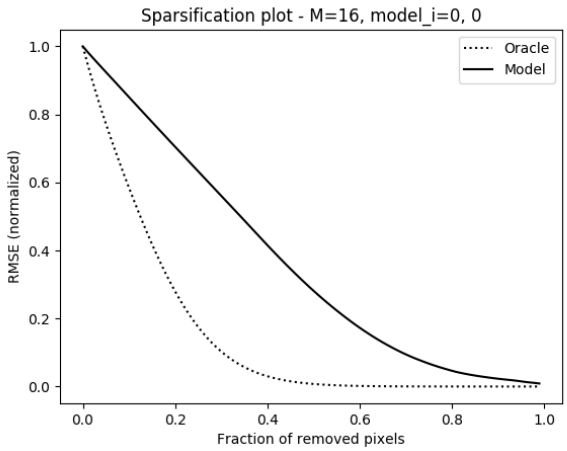
(b) $M = 2$.



(c) $M = 4$.

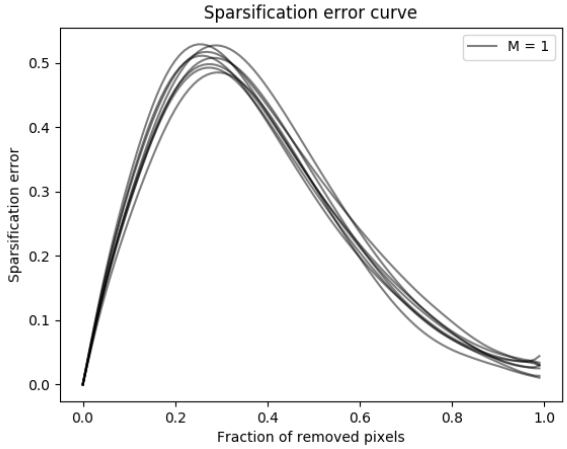


(d) $M = 8$.

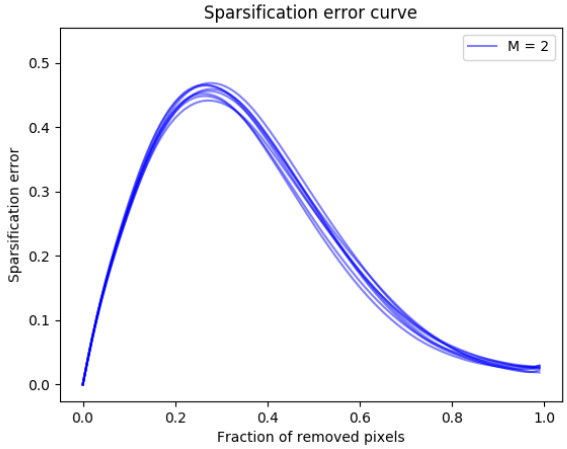


(e) $M = 16$.

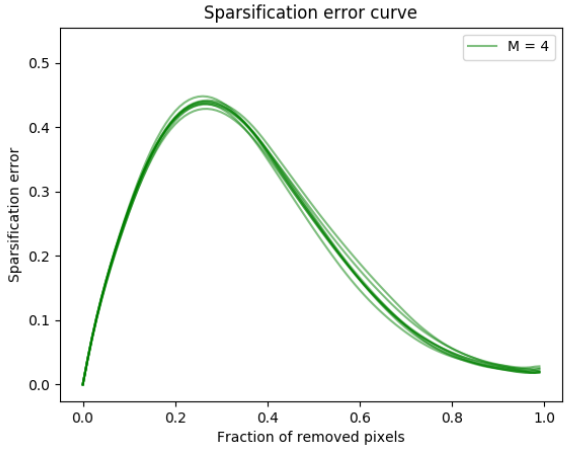
Figure 29: Results for MC-dropout on the Cityscapes validation dataset. Examples of sparsification plots.



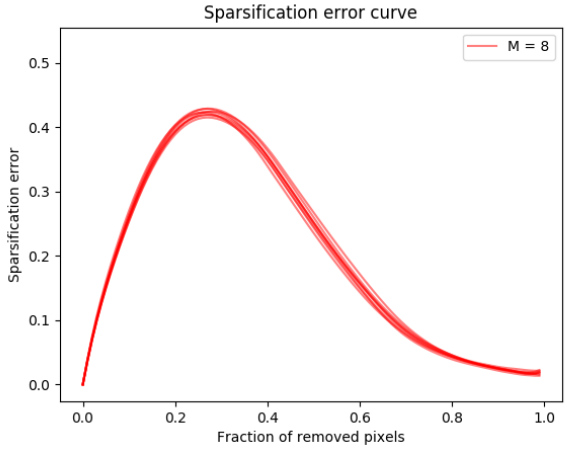
(a) $M = 1$.



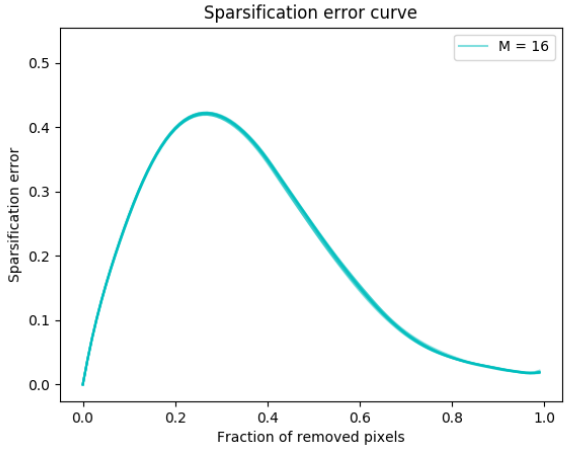
(b) $M = 2$.



(c) $M = 4$.

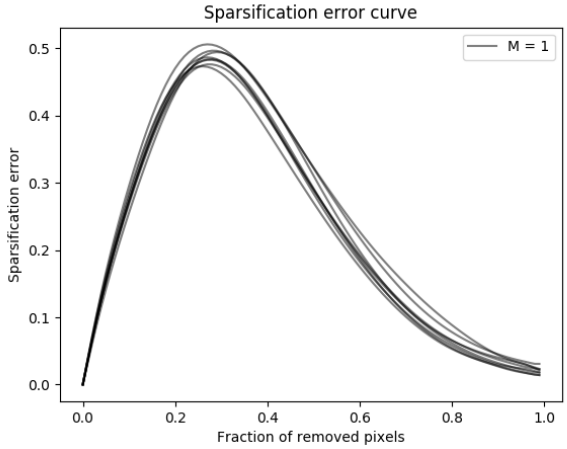


(d) $M = 8$.

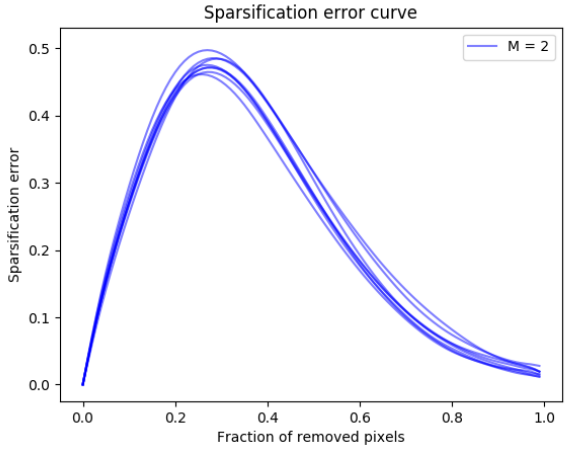


(e) $M = 16$.

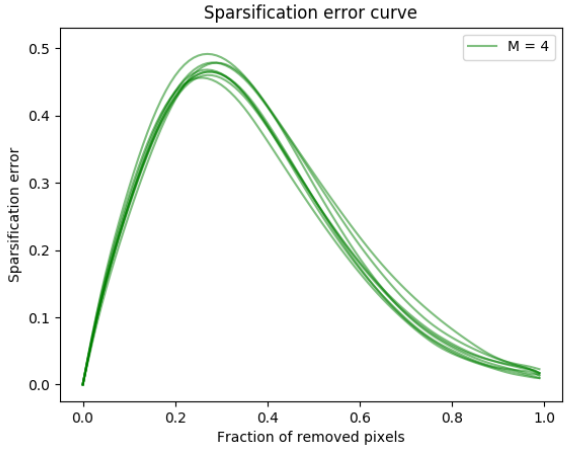
Figure 30: Results for ensembling on the Cityscapes validation dataset. Condensed sparsification error curves.



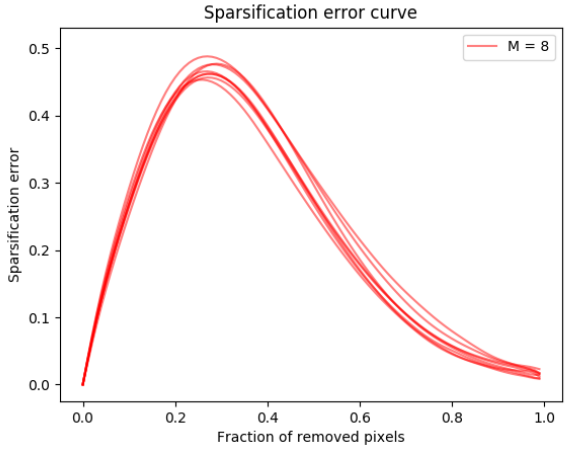
(a) $M = 1$.



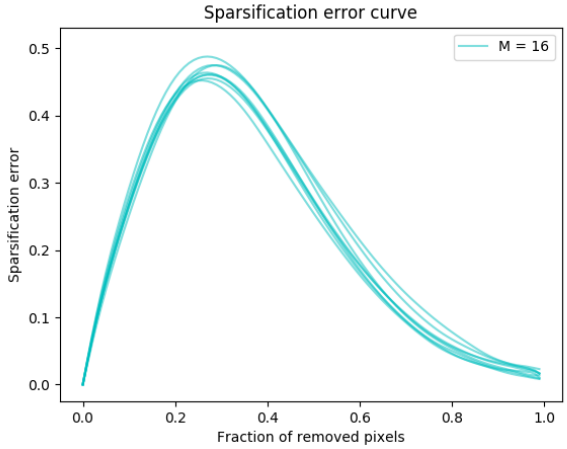
(b) $M = 2$.



(c) $M = 4$.

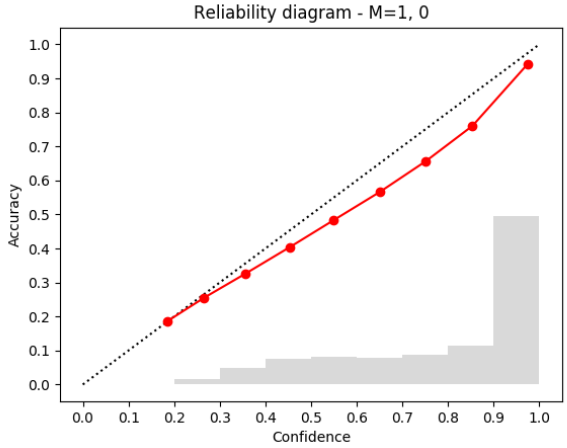


(d) $M = 8$.

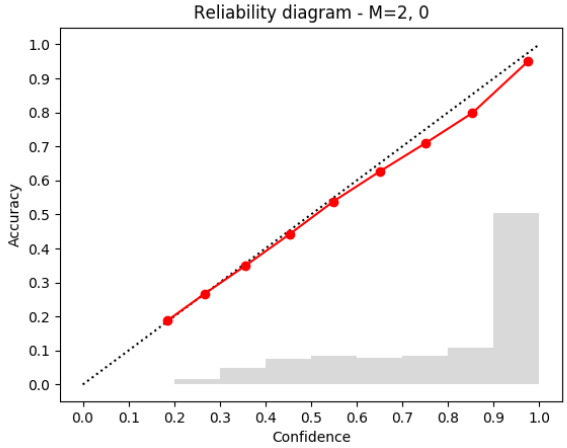


(e) $M = 16$.

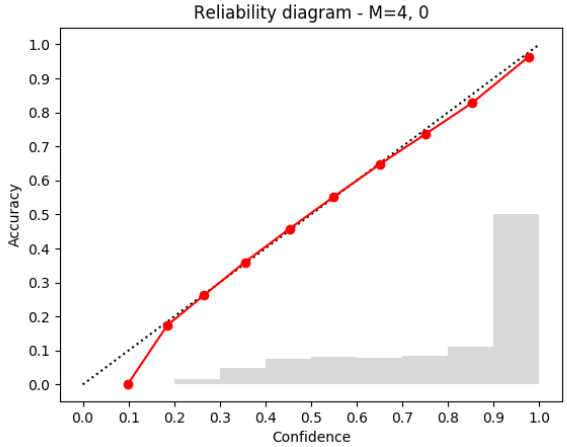
Figure 31: Results for MC-dropout on the Cityscapes validation dataset. Condensed sparsification error curves.



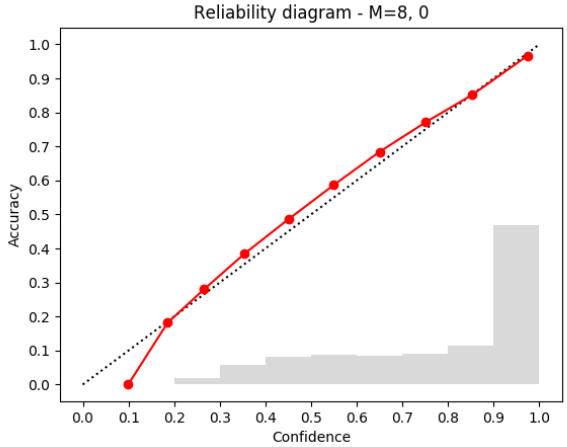
(a) $M = 1$.



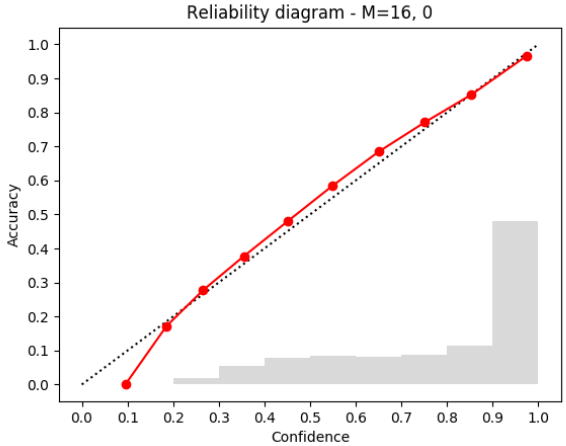
(b) $M = 2$.



(c) $M = 4$.

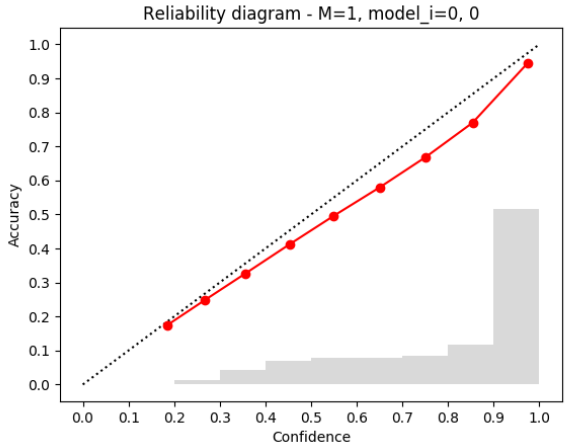


(d) $M = 8$.

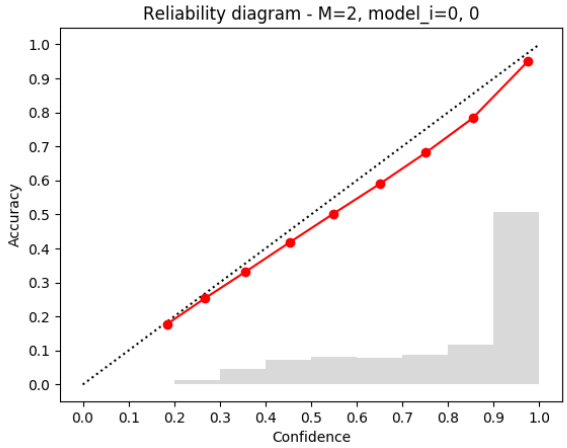


(e) $M = 16$.

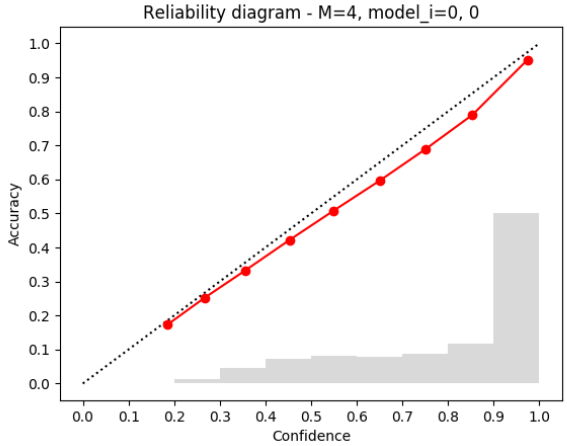
Figure 32: Results for ensembling on the Cityscapes validation dataset. Examples of reliability diagrams with histograms.



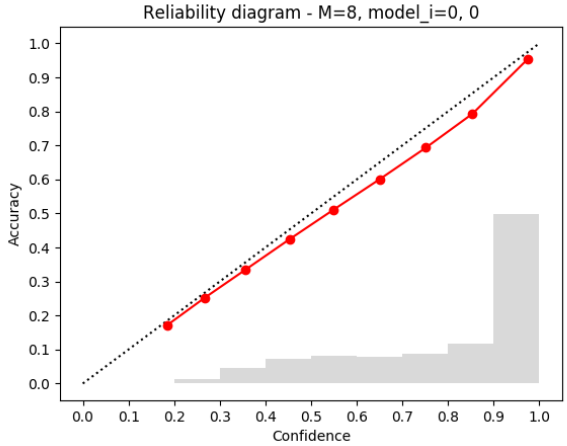
(a) $M = 1$.



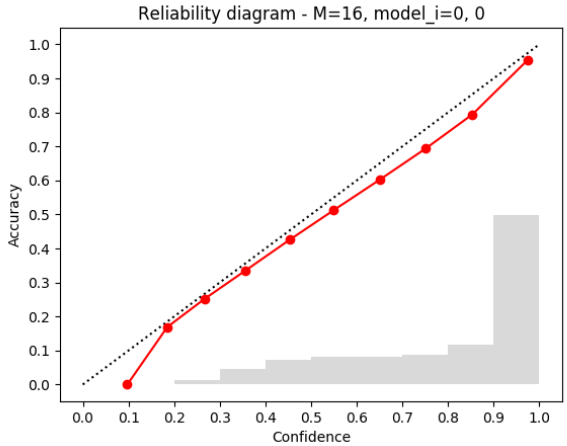
(b) $M = 2$.



(c) $M = 4$.

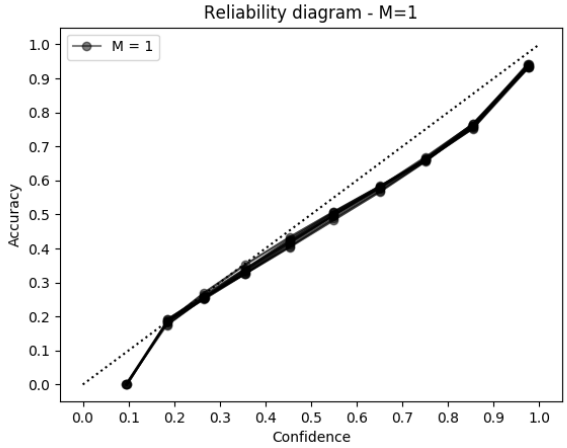


(d) $M = 8$.

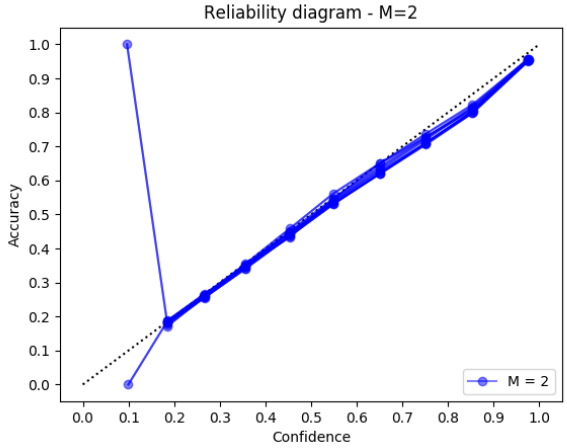


(e) $M = 16$.

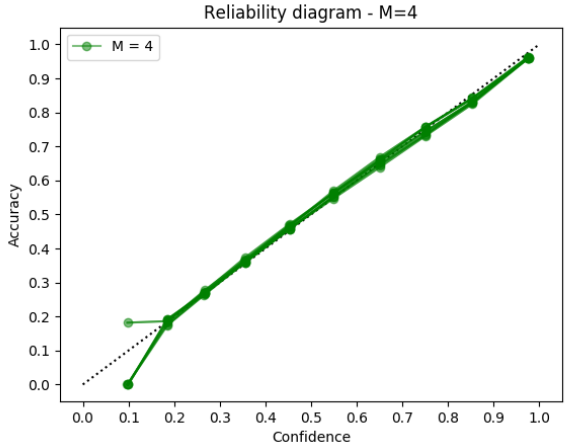
Figure 33: Results for MC-dropout on the Cityscapes validation dataset. Examples of reliability diagrams with histograms.



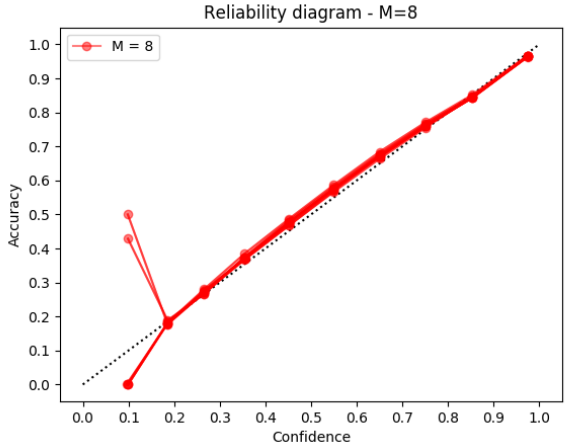
(a) $M = 1$.



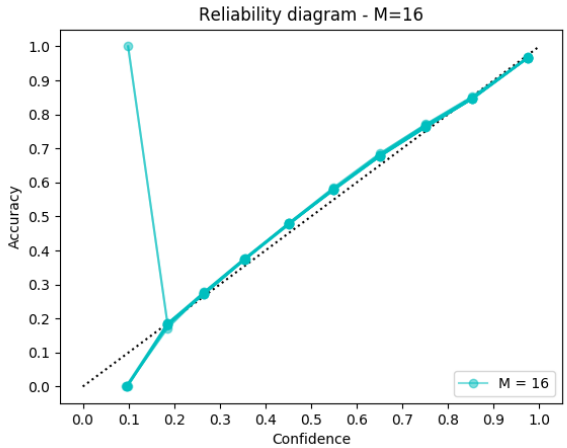
(b) $M = 2$.



(c) $M = 4$.

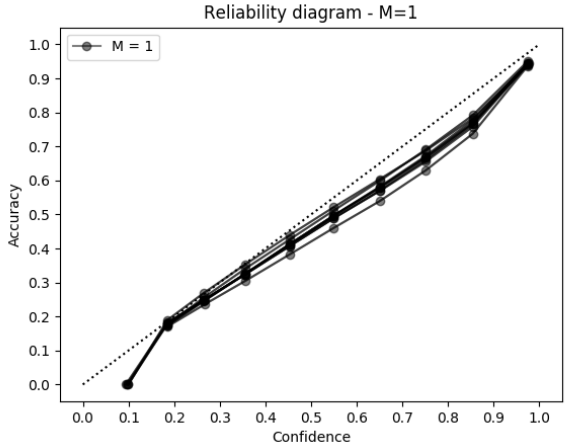


(d) $M = 8$.

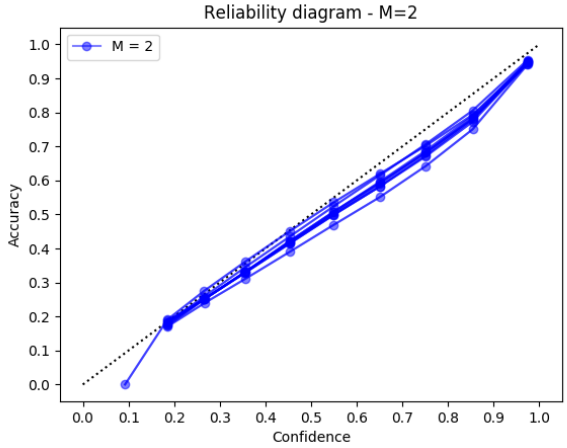


(e) $M = 16$.

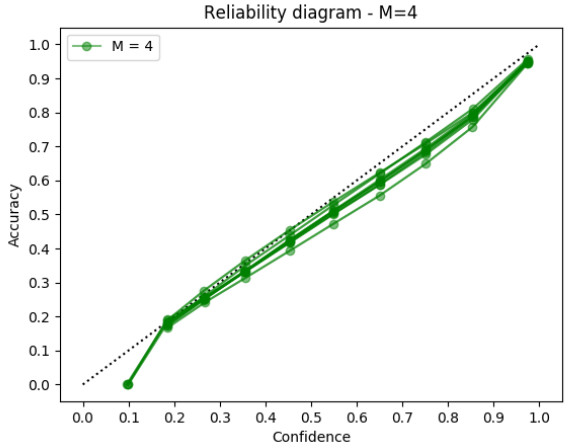
Figure 34: Results for ensembling on the Cityscapes validation dataset. Condensed reliability diagrams.



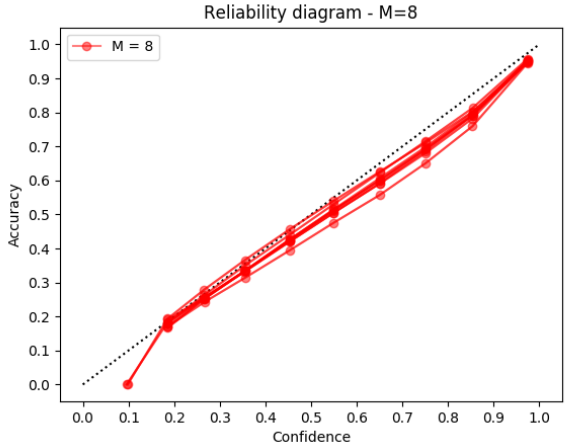
(a) $M = 1$.



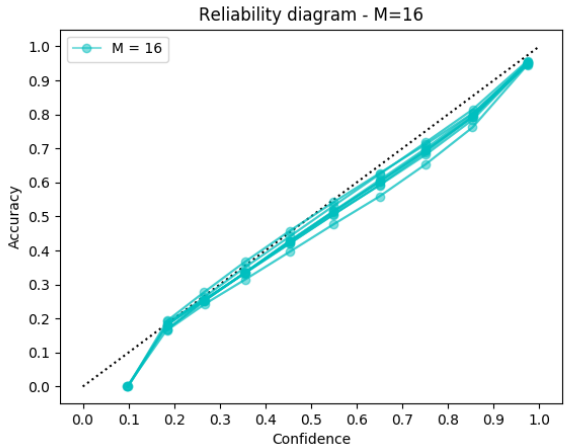
(b) $M = 2$.



(c) $M = 4$.



(d) $M = 8$.



(e) $M = 16$.

Figure 35: Results for MC-dropout on the Cityscapes validation dataset. Condensed reliability diagrams.