## Supplementary material

### a) Reverse Huber (berHu) likelihood

The reverse Huber (berHu) loss is defined as follows:

$$\ell(y) = |y|\mathcal{I}(|y| \leq c) + \left(\frac{y^2 + c^2}{2c}\right)\mathcal{I}(|y| > c) \tag{13}$$

Where $\mathcal{I}(.)$ is the indicator function and $c > 0$ is an appropriately chosen threshold. $\ell(y)$ yields a balance between $\mathcal{L}_1$ and $\mathcal{L}_2$ losses: for smaller residuals, $\mathcal{L}_1$ is considered in order to yield gradients with larger magnitudes, whereas the $\mathcal{L}_2$ component provides an increased penalty to large residuals so that the network also accounts for these.

In this case, it can be shown that $Z_0 = 2\left(1 - e^{-\mathrm{c}} + e^{-\mathrm{c}/2}(2\pi\mathrm{c})^{1/2}\Phi(-c^{1/2})\right)$, where $\Phi(.)$ is the standard normal CDF, and taking $c \to \infty$ recovers the Laplace distribution. Given a new input $x^*$, the predictive mean $m(x^*)$ and variance $\mathbb{V}(x^*)$ can be written as follows:

$$\begin{aligned} m(x^*) &= m\left(q(f(x^*))\right), \\ \mathbb{V}(x^*) &= \underbrace{w(c)\sigma^2(x^*)}_{\text{aleatoric}} + \underbrace{\mathbb{V}(q(f(x^*))}_{\text{epistemic}}, \\ w(\mathrm{c}) &= \frac{-4(\mathrm{c}+1)e^{-\mathrm{c}} + 4 + 2e^{-\mathrm{c}/2}(2\pi)^{1/2}(\mathrm{c})^{3/2}\Phi(-c^{1/2})}{Z_0} \end{aligned} \tag{14}$$

$m(x^*)$ is the mean of the variational distribution $q(f(x^*))$, which follows from the law of conditional expectation, and the decomposition of its variance as sum of epismetic and aleatoric components follows from the law of total variance. Compared to choosing gaussian or laplacian likelihoods, berHu yields a weighted version of $\sigma^2(x^*)$ which depends on the choice of $c$. We choose $c = \frac{1}{5}\max_i \mathbb{E}_{q(f(x_i))}\left(|y_i - f(x_i)|\right)$ during training, where $i$ indexes all output feature maps in a mini-batch and the inner expectation is replaced by a monte carlo estimate. In order to compute the predictive distribution at test time, we record the maximum value of $c$ across all batches in the final training epoch.

### b) Bayesian CNN GP prior

In Figure 4 we display the Bayesian CNN architecture, with weight prior variance of $0.2$ and bias' prior variance of $0.08$, using relu activations, from which the equivalent kernel was derived. This covariance kernel encompasses the behaviour of this Bayesian CNN architecture in the limit where the number of channels in its hidden layers, $C$, tends to infinity, which we denote as Bayesian CNN GP prior. The red and gray blocks correspond to linear interpolation followed by a convolution layer. The sequence of output resolutions for linear interpolation are $20, 40, 60, 80, 100$ percent of the desired output resolution.

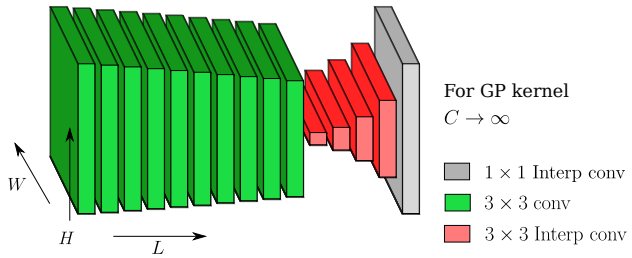


Figure 4. Architecture of the Bayesian CNN GP prior.

### c) Semantic segmentation on CamVid

During training, for all methods, we augment the CamVid dataset by performing random horizontal flips with $0.5$. For our approach (Ours-Boltzmann) we estimate the expected log-likelihood term using $20$ monte carlo samples from the variational distribution.

We select and discuss two test cases in order to compare our method (Ours-Boltzmann) and MCDropout-Boltzmann. Unknown segmentation classes have been masked out as yellow in the plots corresponding to ground truth and predicted

classes. In Figure 5, MCDropout-Boltzmann (top) is wrongly overconfident that the left sidewalk is part of the road, while ours correctly accounts for this difference by outputting higher predictive entropy. Figure 6 displays a failure test case from our method, in which it displays high confidence (low-entropy) that the bus-stop is part of the housing lots. MCDropout-Boltzmann does a better job at flagging out this unknown segmentation class by outputting higher entropy on several of its regions.

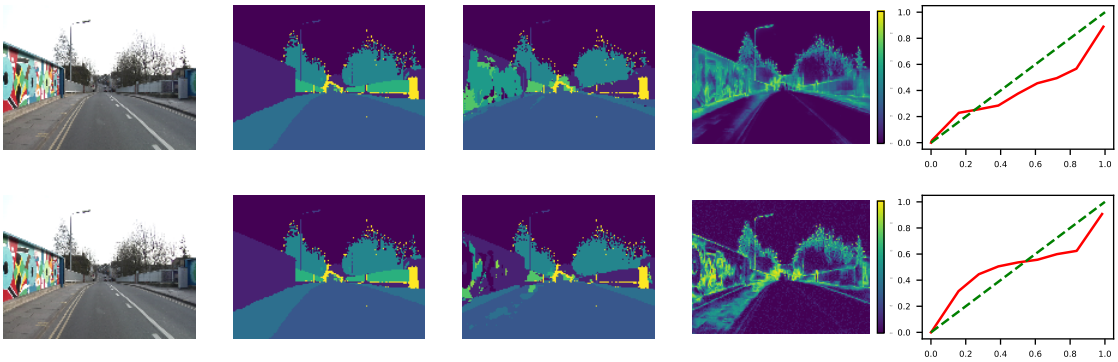

Figure 5. Semantic segmentation on CamVid. MCDropout-Boltzmann (top) and Ours-Boltzmann (bottom). From left to right: rgb input, ground truth, predicted, entropy, calibration plot

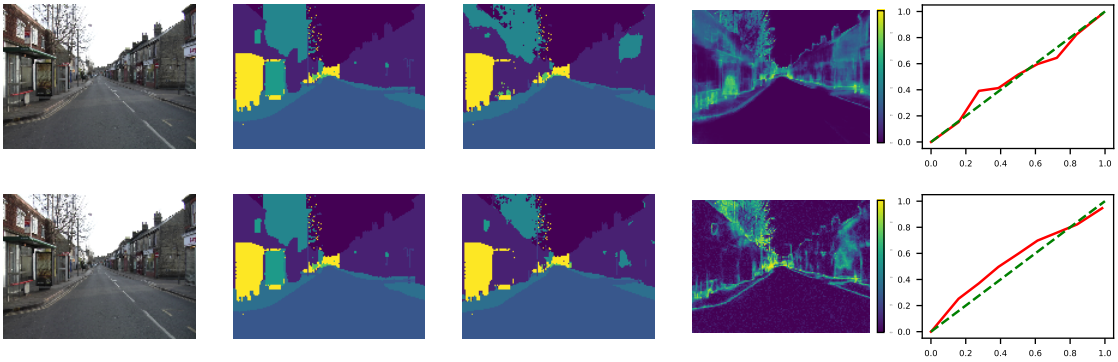

Figure 6. Semantic segmentation on CamVid. MCDropout-Boltzmann (top) and Ours-Boltzmann (bottom). From left to right: rgb input, ground truth, predicted, entropy, calibration plot

## d) Depth estimation on Make3d

During training, for all methods, we augment the Make3d dataset with random horizontal flips (with probability $0.5$), and randomly adjust brightness, saturation, contrast and hue of rgb inputs by a factor of $0.1$. For Ours-Laplace and Ours-berHu, we estimate the expected log-likelihood term using $50$ monte carlo samples from the variational distribution.

We select and discuss two test cases in order to compare our methods (Ours-Laplace, Ours-Gaussian and Ours-berHu) with MCDropout-Laplace. In Figure 7 we display an example where all methods perform well in terms of the predicted depth map. We can observe that both the predicted depth maps and uncertainty from our methods have a sharper aspect than MCDropout-Laplace, which we have consistently observed for most predictions. In Figure 8 we display a failure case for all methods, in terms of predicting inaccurate depth maps. Predictive uncertainty, both its epistemic and aleatoric components, is expected to be higher around the blue sky region. Our methods deliver this effect, while MCDropout-Laplace is overconfident about the predicted depth maps in this region.

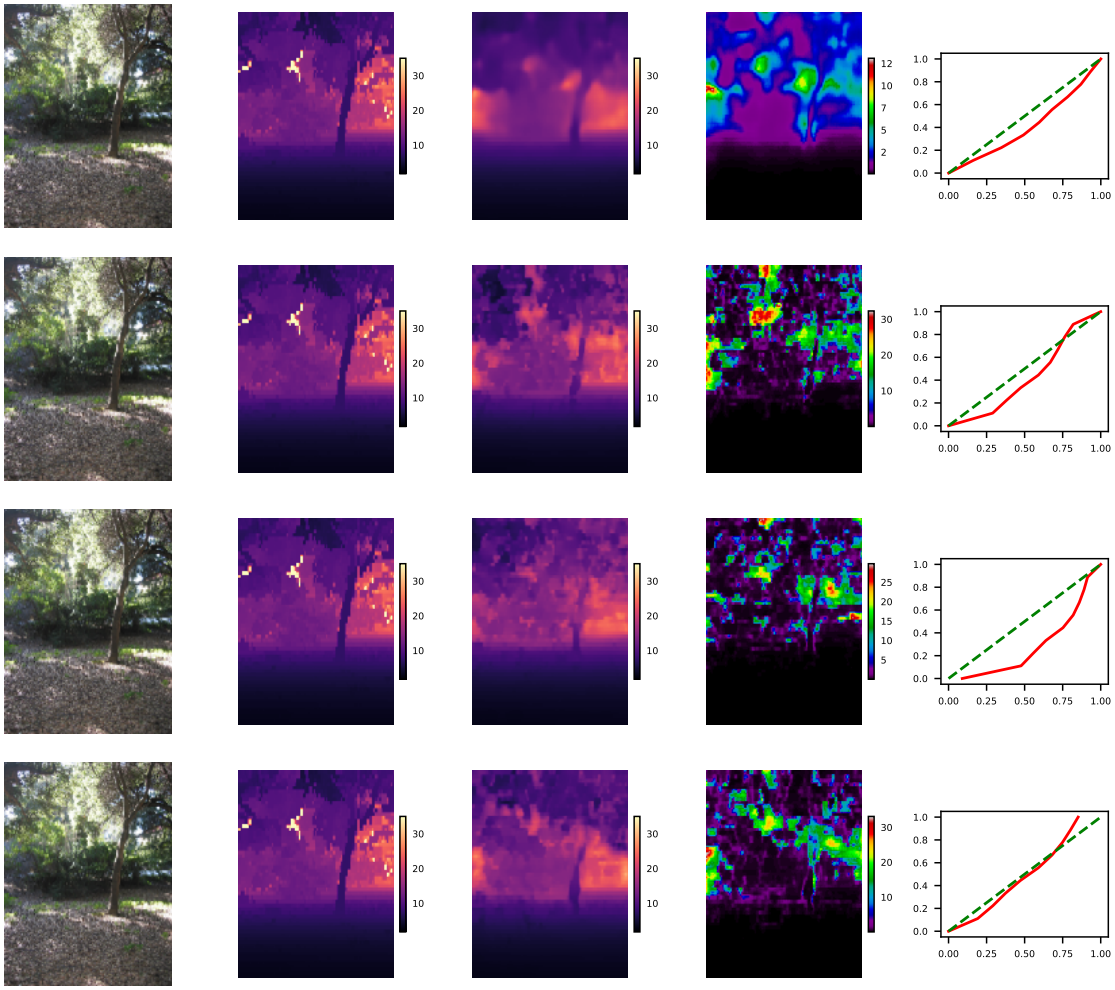


Figure 7. Depth estimation on Make3d. MCDropout-Laplace (first row), Ours-Laplace (second row), Ours-Gaussian (third row), Ours-berHu (fourth row). From left to right: rgb input, ground truth, predictive mean, predictive standard deviation, calibration plot.

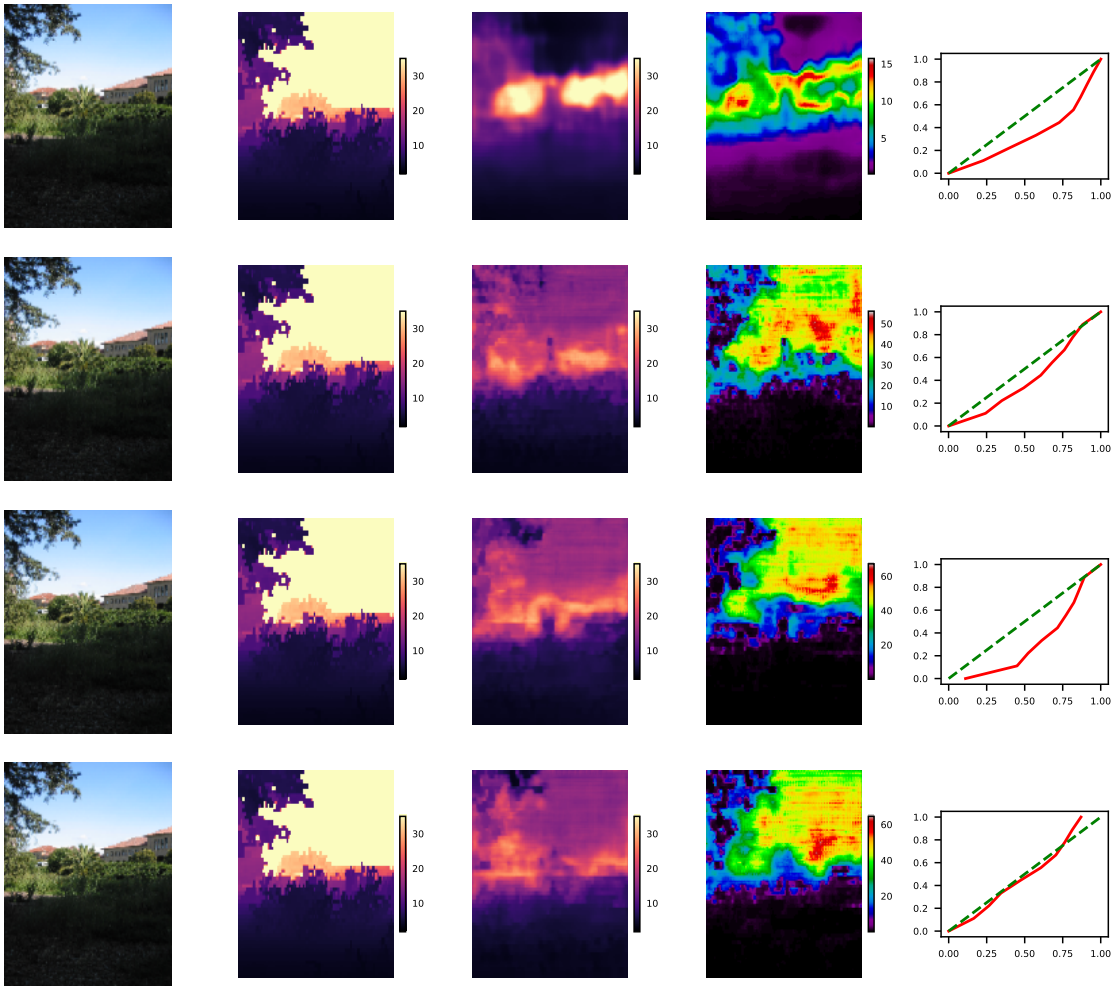


Figure 8. Depth estimation on Make3d. MCDropout-Laplace (first row), Ours-Laplace (second row), Ours-Gaussian (third row), Ours-berHu (fourth row). From left to right: rgb input, ground truth, predictive mean, predictive standard deviation, calibration plot.