Notes on the Behavior of MC Dropout

Francesco Verdoja Ville Kyrki

Abstract

Among the various options to estimate uncertainty in deep neural networks, Monte-Carlo dropout is widely popular for its simplicity and effectiveness. However the quality of the uncertainty estimated through this method varies and choices in architecture design and in training procedures have to be carefully considered and tested to obtain satisfactory results. In this paper we present a study offering a different point of view on the behavior of Monte-Carlo dropout, which enables us to observe a few interesting properties of the technique to keep in mind when considering its use for uncertainty estimation.

1 Introduction

The increasing interest in the deployment of deep learning solutions in real safety-critical applications ranging from hearthcare to robotics and autonomous vehicles is making apparent the importance to properly estimate the uncertainty of the predictions made by deep neural networks [1, 2].

While most common neural network architectures only provide point estimates, uncertainty can be evaluated with Bayesian neural networks (BNNs) [3, 4] where the deterministic weights used in the majority of neural networks are replaced with distributions over the network parameters. Although the formulation of BNNs is relatively easy in theory, their use in practise for most complex problems is often unfeasible due to their need to analytically evaluate the marginal probabilies during training which becomes quickly intractable. Recently, variational inference methods have been proposed as a practical alternative to BNNs, but most of these formulations requires double the number of parameters of a network to represent its uncertainty which leads to increased computational costs [5, 6].

Another very popular option to model uncertainty in deep neural networks is the use of dropout as a way to approximate Bayesian variational inference [6]. The simplicity of the key idea of this formulation is one the main reasons for its popularity: by enabling dropout not only in training but also during testing, and by doing several forward passes through the network with the same input data,

This work was supported by the Strategic Research Council at Academy of Finland, decision 314180. F. Verdoja and V. Kyrki are with School of Electrical Engineering, Aalto University, Finland. {first.surname}@aalto.fi

one can use the distribution of the outputs of the different passes to estimate the first two moments (mean and variance) of the predictive distribution. The mean is then used as the estimate, and the variance as a measure of its uncertainty. This technique, called Monte-Carlo dropout (MCD), has proved effective to, *e.g.,* increase visual relocalization accuracy [7] and semantic segmentation performance on images [1]. Despite its success and simplicity however, it has been recognized that the quality of the uncertainty estimates is tied to parameter choices which need to be calibrated to match the uncertainty [8, 9, 10, 11]. However, when using MCD in practical applications, architectural choices like where to insert the dropout layers, how many to use, and the choice of dropout rate are often either empirically made or set a priori [1, 12, 13], leading to possibly suboptimal performance.

In this work, we conduct a study providing some observations both in theory and through experiments over the behavior of MCD. The main contributions of this work are a theoretical analysis over the behavior of MCD on a simple single-layer linear network, extending and correcting the discussion in [14], and an experimental demonstration that the properties found in theory apply to bigger non-linear networks as well. In the discussion, we offer different intuitions over architecture design and training choices for networks using MCD for uncertainty estimation.

2 Behavior of MC Dropout

In this section we expand and correct the intuitions first presented in [14] over the behavior of MCD. To this end, let us consider a single-layer linear network

$$f = \sum_{k=1}^{K} d_k w_k \tag{1}$$

with weights $w_k \in \mathbb{R}$ and dropouts $d_k \sim \mathrm{Ber}(p)$, *i.e.,* a Bernoulli distribution with success probability p. To note that here p refers to $P(d_k = 1)$, while most dropout implementations require a dropout probability parameter $p_d = P(d_k = 0) = 1 - p$.

Assuming all weights to converge to the same value w, which is to be expected when using dropout, then

$$\begin{aligned} \mathbb{E}\left[f\right] &= \mathbb{E}\left[w \sum_{k=1}^{K} d_k\right] = w\mathbb{E}\left[\mathrm{B}(K,p)\right] = \\ &= wKp \\ \mathrm{Var}\left[f\right] &= \mathrm{Var}\left[w \sum_{k=1}^{K} d_k\right] = w^2 \,\mathrm{Var}\left[\mathrm{B}(K,p)\right] = \\ &= w^2 Kp(1-p) \end{aligned} \tag{2}$$

where $\mathrm{B}(K,p)$ is a binomial distribution with K trials and success probability p.

Given a ground-truth $\{y_1, \dots, y_n\}$ with average $\bar{y} \coloneqq \sum_{i=1}^{n} y_i/n$, minimizing the mean squared error (MSE) from the ground-truth means finding the minimum of

$$\begin{aligned}
\mathbb{E}\left[(f-\bar{y})^2\right] &= \mathbb{E}\left[f^2 - 2f\bar{y} + \bar{y}^2\right] \\
&= \mathbb{E}\left[f^2\right] - 2\bar{y}\mathbb{E}\left[f\right] + \bar{y}^2 \\
&= \mathrm{Var}\left[f\right] + \mathbb{E}^2\left[f\right] - 2\bar{y}\mathbb{E}\left[f\right] + \bar{y}^2 \\
&= w^2Kp(1-p) + w^2K^2p^2 - 2\bar{y}wKp + \bar{y}^2 \\
&= w^2Kp(Kp-p+1) - 2\bar{y}wKp + \bar{y}^2
\end{aligned} \tag{3}$$

This can be achieved by solving

$$\begin{aligned}
\frac{d}{dw}\mathbb{E}\left[(f-\bar{y})^2\right] &= 0 \\
2wKp(Kp-p+1) - 2\bar{y}Kp &= 0 \\
w(Kp-p+1) - \bar{y} &= 0
\end{aligned} \tag{4}$$

which means the optimal weight is

$$w = \frac{\bar{y}}{Kp-p+1} = \frac{\bar{y}}{K(1-p_d)+p_d} \tag{5}$$

Consequently, combining Eq. (2) and Eq. (5), we find that at convergence:

$$\begin{aligned}
\mathbb{E}\left[f\right] &= \frac{Kp\bar{y}}{Kp-p+1} = \frac{K(1-p_d)\bar{y}}{K(1-p_d)+p_d} \\
\mathrm{Var}\left[f\right] &= \frac{Kp(1-p)\bar{y}^2}{(Kp-p+1)^2} = \frac{Kp_d(1-p_d)\bar{y}^2}{(K-Kp_d+p_d)^2}
\end{aligned} \tag{6}$$

From Eq. (6), a few observation can be drawn:

1. while one would expect $\mathbb{E}\left[f\right] = \bar{y}$, the expected output of the network is actually introducing a bias. For big network however, this bias is negligible, since $Kp \approx Kp - p + 1$;

2. the size of the variance of the posterior distribution generated by MCD on this simple network depends on the interaction between the dropout rate p_d and the model size K;

3. the posterior distribution has no dependence on the amount of data n, nor the observed variance in the data, which means that it does not concentrate as more data is gathered;

4. finally, the variance of the posterior distribution is proportional to $\bar{y}^2$, *i.e.,* the bigger the value to be estimated, the bigger the estimated model uncertainty.

Table 1: Comparison of single-layer network experiment results versus the thoeretical expectation from Eqs. (5) and (6)

		Thoeretical		Experimental	
p_d	Dataset	w	$\mathrm{Var}[f]$	w	$\mathrm{Var}[f]$
0.2	$\mathbf{Y}'$	0.025	0.050	0.025	0.058
0.2	$\mathbf{Y}''$	0.025	0.050	0.026	0.076
0.5	$\mathbf{Y}'$	0.040	0.199	0.040	0.214
0.5	$\mathbf{Y}''$	0.040	0.199	0.040	0.249

While we demonstrate here in theory that these properties of MCD exist on a very simple network, proving them following a similar thoeretical process on bigger networks becomes quickly unfeasible.

For this reason, to try to understand how these interaction work on bigger more realistic networks, we run different experiments, which we will present in the following section.

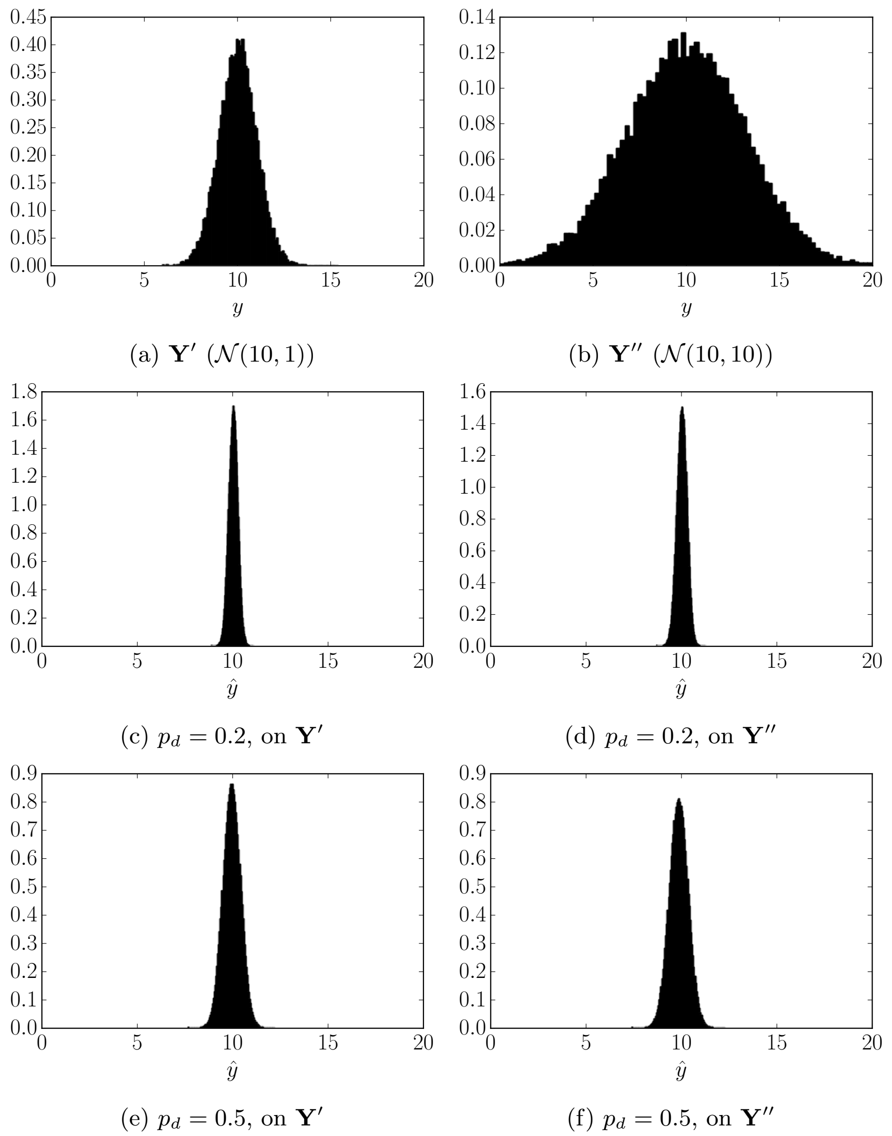
3 Experiments

To verify that the properties found in Section 2 hold true even for bigger non-linear networks, we created the suite of experiments we report in this section. These experiments have been implemented in *pytorch v0.3.0*. The source code to replicate these experiment will be made available in the form of jupyter notebooks before the workshop start date.

3.1 Single-layer linear network

In this first experiment we want to empirically verify the theoretical behavior observed in Section 2. In particualar, we implemented two versions of the single-layer linear network in Eq. (1) with $K = 500$, one with $p_d = 0.2$ and one with $p_d = 0.5$ respectively. Once more, note that the dropout probability $p_d = 1 - p$. We created two ground-truth datasets $\mathbf{Y}' = \{y'_1, \ldots, y'_n\}$ and $\mathbf{Y}'' = \{y''_1, \ldots, y''_n\}$ with $n = 3200$, $y'_i \sim \mathcal{N}(10, 1)$, and $y''_i \sim \mathcal{N}(10, 10)$. We trained both networks on both datasets for 600 epochs with MSE loss and adam optimizer. As a side note, training with MSE holds the same results as training with log-likelihood, since the minimum of the two losses is in the same position in this case. After training, we run multiple forward passes of the networks with dropout active to obtain one million samples to empirically estimate $\mathrm{Var}[f]$.

In Table 1, we can see that the values for w and $\mathrm{Var}[f]$ found experimentally match the theoretical ones computed using Eqs. (5) and (6), aside for noise introduced by the sampling process. Moreover, they confirm how changing the parameter p_d afftects the model uncertainty, while changing observed variance in the data (from $\sigma = 1$ in $\mathbf{Y}'$, to $\sigma = 10$ in $\mathbf{Y}''$) does not. This phenomenon is also clearly visible in Fig. 1 where it can be noticed how the variance of the


(a) $\mathbf{Y}'$ ($\mathcal{N}(10, 1)$)

(b) $\mathbf{Y}''$ ($\mathcal{N}(10, 10)$)

(c) $p_d = 0.2$, on $\mathbf{Y}'$

(d) $p_d = 0.2$, on $\mathbf{Y}''$

(e) $p_d = 0.5$, on $\mathbf{Y}'$

(f) $p_d = 0.5$, on $\mathbf{Y}''$

Figure 1: Ground-truth distributions (a, b) and corresponding MCD output distributions for single-layer linear networks with $p_d = 0.2$ (c, d) and $p_d = 0.5$ (e, f) respectively.

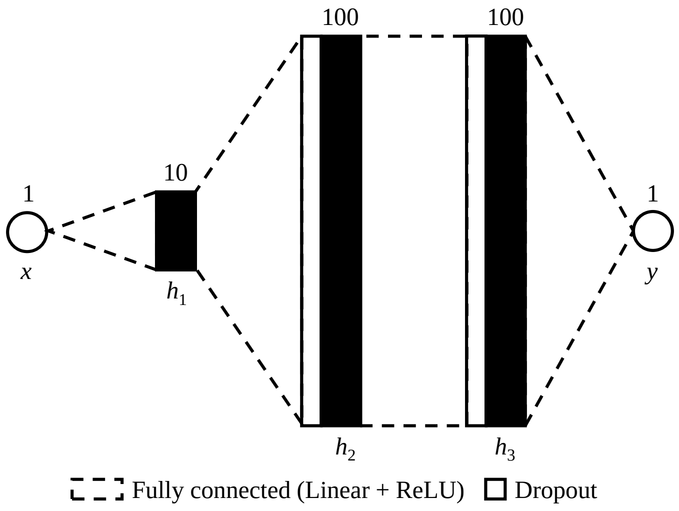


Figure 2: Fully connected network used in the experiments.

output distribution of MCD is impacted by p_d (compare Figs. 1c and 1e, and Figs. 1d and 1f) while being unaffected by the variance in the dataset it trained on (compare Figs. 1c and 1d, and Figs. 1e and 1f).

3.2 Non-linear networks

To verify that effects similar to those presented in the previous section can still be observed after introducing more layers and non-linearities, we performed a similar experiment on the fully connected neural network shown in Fig. 2. While this network is still smaller then most real networks, we decided for this size because it had enough complexity while still being simple enough to analyze. In all the following experiments we trained for 1000 epochs with MSE loss and adam optimizer on a dataset of 32000 samples. We trained two variants of the network with dropout rate $p_d = 0.2$ and $p_d = 0.5$ respectively. As it can be seen in the figure, for these experiments the network has one input x and produces one output y, effectively approximating a function $f : \mathbb{R} \to \mathbb{R}$.

Fig. 3 shows the results produced by the network on a noisy constant function where for each input $x \in [0, 1]$ multiple possible outputs y are present in the dataset while $\forall x, \mathbb{E}[y] = 0.5$ (Fig. 3a). We first show the behavior of the network when the last linear layer has a bias term in Fig. 3b. It can be seen that the network is nullifing the variance completely. It is indeed setting all weights to 0 and encoding the desired (constant) output in the bias. If we remove the bias from the last linear layer (Figs. 3c and 3d) we obtain constant variance, proportional to the dropout rate.

Fig. 4 shows another experiment, conducted on different functions. In this case the datasets have no noise, *i.e.,* for each x, all samples in the dataset have

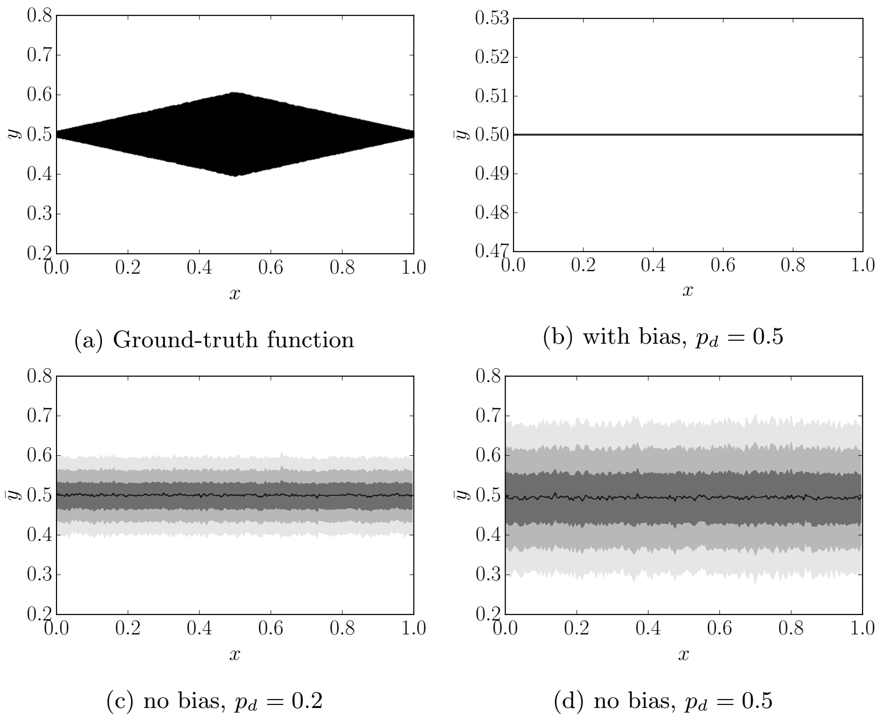
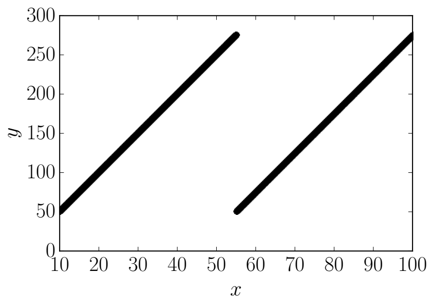
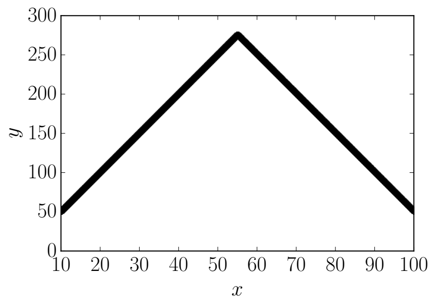


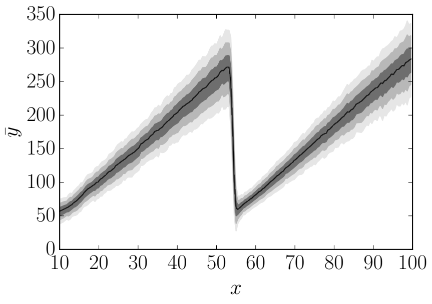
Figure 3: Results obtained by training a non-linear network on the dataset shown in (a); different variants of the network have been trained: one with a bias term in the last linear layer (b) and two without bias in the last linear layer and with different dropout rates (c, d). In (a) each dot represents a datapoint; in (b, c, d) the line represents the average output of 300 forward passes through the network, with the shaded areas representing σ, 2σ, and 3σ respectively.



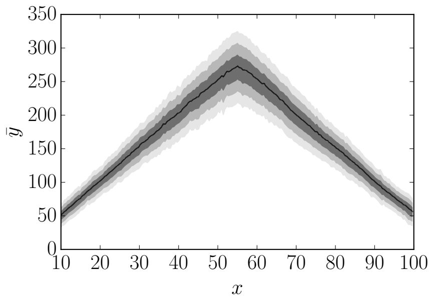
(a) Ground-truth function



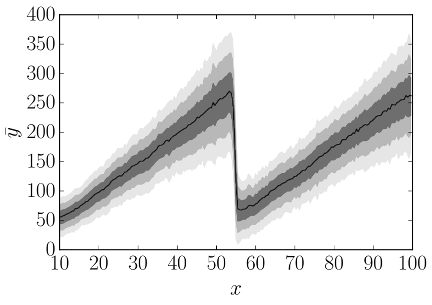
(b) Ground-truth function



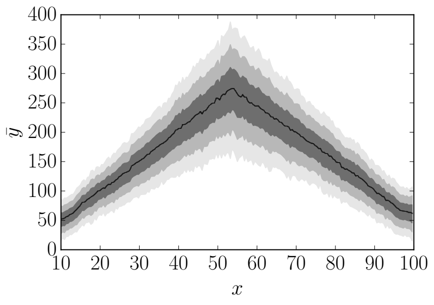
(c) $p_d = 0.2$



(d) $p_d = 0.2$



(e) $p_d = 0.5$



(f) $p_d = 0.5$

Figure 4: Results obtained by training a non-linear network on the dataset shown in (a, b); two variants of the network have been trained: with dropout rate $p_d = 0.2$ (c,d) and $p_d = 0.5$ (e, f). In (a, b) each dot represents a datapoint; in (c, d, e, f) the line represents the average output of 300 forward passes through the network, with the shaded areas representing σ, 2σ, and 3σ respectively.

the same corresponding y. All the functions and network variants we tested are available as additional material, however, due to space limitations, in here we present only two functions (Figs. 4a and 4b) and only the results for networks where the bias has been disabled on the last linear layer. All results not reported here lead to the same conclusions. From these results, the last of the properties presented in Section 2 can be noticed, namely the fact that the size of the uncertainty scales proportionally to the value of y. These also confirm one more time the dependence of the uncertainty from the dropout parameter p_d.

4 Discussion

The results presented in Section 3 confirm that the behaviors of MCD that we theoretically observed on a single-layer linear network are still present even as the size and complexity of the network grows. When looking at these results, we are able to make a few observations.

First of all, all experiments provided an additional demonstration that the choice of dropout rate p_d is crucial and that in itself the epistemic uncertainty estimate provided by MCD is not affected by the amount of data available during training or its variance, which is a good reminder that the uncertainty estimate produced by MCD is not calibrated and that the the dropout rate has to be adjusted to match the epistemic uncertainty. In practise, this is usually done through grid search, although several methods have been proposed to learn the parameter during training, *e.g.,* [15, 16, 10].

Another interesting aspect that emerged from our study was the scaling of the uncertainty based on the value of the network output. This effect, if not taken into account, can lead to degraded quality of the uncertainty estimates, which we empiracally noticed in the past in a practical robotic application [13]. This effect is particularly visible when dropout is applied before the last linear layer of the network, as is the case in the the experiments presented here. If more non-linear layers are added after the last dropout layer the non-linearities can reduce this effect. This would suggest that in applications with widely varying outputs, it could be useful to limit dropout only to the inner parts of the network, justifying what other studies have found experimentally to helps improve performance [1, 12].

5 Conclusions

In this paper we presented a different point of view on the behavior of MCD that enabled us to show both theoretically and experimentally different properties of that approach, in particular the dependency of the uncertainty estimates from the dropout rate and from the value the network is trying to estimate. Our intent was to propose a framework to demonstrate these property in a very understandable fashion, to help in both architecture design and training procedures when using MCD.

References

[1] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *British Machine Vision Conference 2017, BMVC 2017*, Jan. 2017.

[2] A. Loquercio, M. Segù, and D. Scaramuzza, "A General Framework for Uncertainty Estimation in Deep Learning," *IEEE Robotics and Automation Letters*, vol. 5, pp. 3153–3160, Apr. 2020.

[3] J. S. Denker and Y. LeCun, "Transforming Neural-net Output Levels to Probability Distributions," in *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, NIPS'90, (San Francisco, CA, USA), pp. 853–859, Morgan Kaufmann Publishers Inc., 1990.

[4] D. J. C. MacKay, "A Practical Bayesian Framework for Backpropagation Networks," *Neural Computation*, vol. 4, pp. 448–472, May 1992.

[5] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Network," in *International Conference on Machine Learning*, pp. 1613–1622, June 2015.

[6] Y. Gal and Z. Ghahramani, "Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, vol. 48 of *ICML'16*, (New York, NY, USA), pp. 1050–1059, 2016.

[7] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4762–4769, May 2016.

[8] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep Exploration via Bootstrapped DQN," in *Advances in Neural Information Processing Systems 29 (NIPS 2016)* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 4026–4034, Curran Associates, Inc., 2016.

[9] T. Pearce, N. Anastassacos, M. Zaki, and A. Neely, "Bayesian Inference with Anchored Ensembles of Neural Networks, and Application to Reinforcement Learning," *arXiv:1805.11324 [cs, stat]*, May 2018.

[10] S. Boluki, R. Ardywibowo, S. Z. Dadaneh, M. Zhou, and X. Qian, "Learnable Bernoulli Dropout for Bayesian Deep Learning," *arXiv:2002.05155 [cs, stat]*, Feb. 2020.

[11] J. Caldeira and B. Nord, "Deeply Uncertain: Comparing Methods of Uncertainty Quantification in Deep Learning Algorithms," *arXiv:2004.10710 [physics, stat]*, Apr. 2020.

[12] A. Jungo, R. McKinley, R. Meier, U. Knecht, L. Vera, J. Pérez-Beteta, D. Molina-García, V. M. Pérez-García, R. Wiest, and M. Reyes, "Towards Uncertainty-Assisted Brain Tumor Segmentation and Survival Prediction," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, S. Bakas, H. Kuijf, B. Menze, and M. Reyes, eds.), Lecture Notes in Computer Science, (Cham), pp. 474–485, Springer International Publishing, 2018.

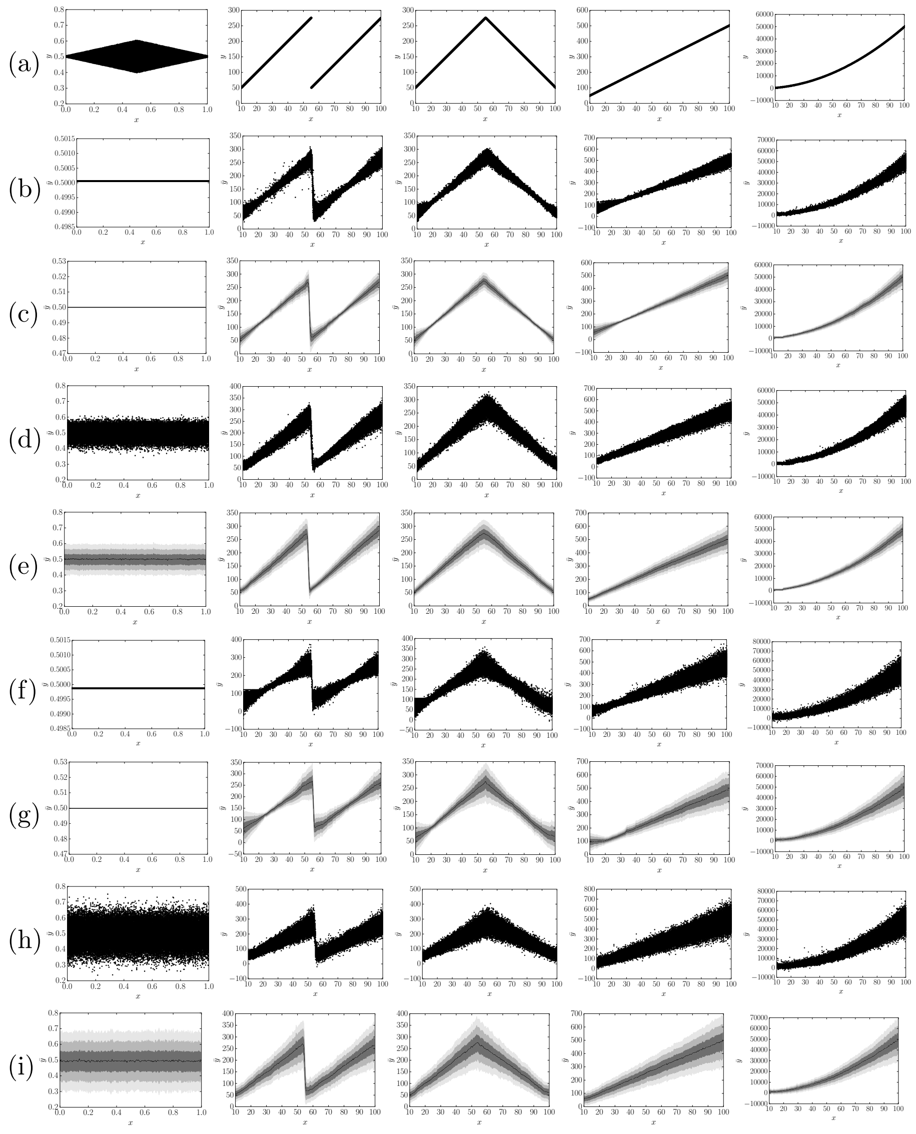
[13] F. Verdoja, J. Lundell, and V. Kyrki, "Deep Network Uncertainty Maps for Indoor Navigation," in *2019 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, (Toronto, Canada), Oct. 2019.

[14] I. Osband, "Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout," in *NIPS 2016 Workshop on Bayesian Deep Learning*, (Barcelona, Spain), Oct. 2016.

[15] Y. Gal, J. Hron, and A. Kendall, "Concrete Dropout," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 3581–3590, Curran Associates, Inc., 2017.

[16] B. Phan, R. Salay, K. Czarnecki, V. Abdelzad, T. Denouden, and S. Vernekar, "Calibrating Uncertainties in Object Localization Task," *arXiv:1811.11210 [cs, stat]*, Nov. 2018.

Additional results obtained by training a non-linear network on different datasets (row a). Different variants of the network have been trained: one with $p_d = 0.2$ and bias term in the last linear layer (rows b, c), one with $p_d = 0.2$ without bias term in the last linear layer (rows d, e), one with $p_d = 0.5$ and bias term in the last linear layer (rows f, g), and one with $p_d = 0.5$ without bias term in the last linear layer (rows h, i). In (rows a, d, h) each dot represents a datapoint; in (rows c, e, g, i) the line represents the average output of 300 forward passes through the network, with the shaded areas representing σ, 2σ, and 3σ respectively.


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
x
y