Safe ML Uncertainty Final Report

Lim, Hojun. Hu, Zhengjiang.
Supervisor: Sicking, Joachim. Akila, Maram. Müller, Emmanuel.

July 2020

1 Introduction

With the development of neural networks, Deep Learning, as a new field, has been witnessed to show a vast development in most of the areas related to machine learning and data science. The capability of deep neural networks to model complex functions have surprised people which becomes later a driven force for the community to shift from traditional approaches to the applications of neural networks to various tasks.

Neural networks are known as being good at learning from large scale data and accomplishing complicate tasks. However, how neural networks reach their predictions, and how the networks are confident of their conclusions remain to be a black box process. Formally speaking, modern neural networks are not designed to be able to capture predictive uncertainty [2, 5]. However, describing model uncertainty draws increasing attention as its significance has proved to be critical in various, especially safety-oriented applications such as Autonomous Driving [7].

In this report, we focus on analyzing and comparing several popular methods for modeling predictive uncertainty and choose MC-Dropout, Deep Ensemble and Parametric Uncertainty as the representatives of such methods. We demonstrate the experiments of applications of the methods to several UCI regression datasets and study the realism of uncertainty [7] described by the methods. Further, it is important for a good model to produce higher uncertainty when it is given unseen data. Therefore, we report the results of MC-Dropout and Parametric Uncertainty applied to "out of distribution" data and analyze their performance. We show that Parametric Uncertainty outperforms MC-Dropout on some of the datasets studied while for other datasets both methods are affected by the characteristics of the datasets.

2 Related work and Technical Background

2.1 Types of Uncertainties

The aleatoric uncertainty and epistemic uncertainty have been two major types of uncertainty in Bayesian modeling [3, 4]. The aleatoric uncertainty, which is also called data-inherent uncertainty, denotes the intrinsic noise in the observations. Noise from sensors could be one example of such uncertainty, and it cannot be resolved by collecting more data. The epistemic uncertainty, on the other hand, is related to the model parameters. This uncertainty arises when the model is not suitably trained due to the lack of training data. Meaning it can be reduced and explained away once enough data is given, and therefore, it is often referred to as model uncertainty.

2.2 Bayesian uncertainty model

Standard deep learning architectures do not allow uncertainty representation in regression tasks since the network output is deterministic. In order for models to capture uncertainties, the standard neural networks have been extended to Bayesian models with assigning distributions to model's weights. With this approach, the model's output becomes stochastic, and the network has a distribution, which is often to be a Gaussian, over the model's output. This leads to a Bayesian neural network model to capture the uncertainty with distribution parameters.

2.3 MC-Dropout

MC-Dropout is known as one popular method for modeling predictive uncertainty due to its simplicity and flexibility. However, its effectiveness of being an approximation of Bayesian neural networks has been proved by Gal et al. to be impressive [2].

They have shown that deep neural networks with arbitrary depth, non-linearity, dropout layer are a mathematically equivalent approximation to the Bayesian inference in deep Gaussian processes. According to Gal, the approximated model's predictive mean(2) and variance(3) can be computed as

$$q(y^*|x^*) = \int p(y^*|x^*, w)q(w)\ dw \tag{1}$$

$$\mathbb{E}_{q(y^*|x^*)}(y^*) \approx \frac{1}{T}\sum_{t=1}^{T} \hat{y}^*(x^*, W_1^t, \cdots, W_L^t) \tag{2}$$

$$Var_{q(y^*|x^*)}(y^*) = \frac{1}{T}\sum_{t=1}^{T} \hat{y}^*(x^*, W_1^t, \cdots, W_L^t)^T \hat{y}^*(x^*, W_1^t, \cdots, W_L^t) \ - \ \mathbb{E}_{q(y^*|x^*)}(y^*)^T\, \mathbb{E}_{q(y^*|x^*)}(y^*) \tag{3}$$

where $q(y^*|x^*)$ is the approximated distribution of true posterior $p(y^*|x^*, w)$, W denotes network weights, and T implies a number of iterations. This approach means that a network can achieve the similar effect as if it is trained like the Bayesian model by simply introducing a dropout layer into a network, producing model's outputs multiple time, e.g. T times then combining those. Such process is referred as to MC-Dropout, and it can be interpreted as an ensemble combination model where the final prediction is determined based on averaged T network outputs.

2.4 Deep Ensemble(DE)

Despite the advantage of MC-Dropout, several drawbacks also exist. For example, the number of forward passes T of the network during prediction has to remain high enough, which can be computationally expensive. Further, Lakshminarayanan et al. point out that the fixed dropout rate is another limitation as it cannot be learned during training. Therefore, they have proposed Deep Ensemble as an alternative to model uncertainty [5]. Unlike MC-Dropout that generates the ensemble from only one trained model using dropout, Deep Ensemble individually trains multiple models with different parameter initialization. Those models are treated as uniformly-weighted mixture model and combine network outputs to produce the final predictions like bagging(bootstrapping).

2.5 Parametric Uncertainty(PU)

While MC-Dropout and Deep Ensemble methods implicitly model the uncertainty, another uncertainty mechanism: Parametric Uncertainty, which explicitly measures the uncertainty without sample statistics using the negative log likelihood loss function, has been proposed [5].

3 Uncertainty methods on UCI datasets

In order to study model uncertainty, the first step is to reproduce the methods commonly used for uncertainty estimation. Among the majority of such techniques, MC-Dropout [2], Deep Ensembles and Parametric Uncertainty [5] are popular choices that often occur in many researches. In this section, how we used these three methods for modeling uncertainty is explained in detail. We demonstrate the reproduction process of these methods and compare our results with those reported in [5] on several UCI regression datasets [1]. For better comparison, we use the same datasets as used in their paper and meanwhile we restrict all the datasets to have 1D target feature.

3.1 Reproduction of three uncertainty methods

3.1.1 Reproduction of MC-Dropout

To reproduce the results of most datasets reported in [5] we build a neural network with one hidden layer of 50 nodes and apply ReLU as the non-linearity on top of the hidden layer. Exceptionally, the dimension of the hidden layer has been doubled specifically for Year and Protein datasets as these two datasets are relatively large. Subsequently, a dropout layer with dropout rate 0.1 is applied before the final output layer.

During training, we normalize both the input and target features so that the distribution of the training data has zero mean and unit standard deviation. To enforce a smooth and trainable process, Adam has been used as the optimizer with a learning rate of 0.1 and the network is trained for 40 epochs with mini-batch of size 100 on each dataset using the mean squared error (MSE) $\sum_{n=1}^{N}(y_n - \mu(x_n))^2$.

During the evaluation, the dropout is still applied to the network as it does during training. For each input, the network performs, in our study, 50 forward passes with the same dropout rate to generate a sample of the network outputs. The mean and standard deviation of the sampled are treated as the network prediction and its uncertainty, respectively. Both the mean and standard deviation are de-normalized to match the scale of targets of the test dataset. The network performance is tested using negative log-likelihood (NLL) and root mean square error (RMSE). To better evaluate the results, a regularization which we refer to as 'v-noise' is added to the variance of the sample to avoid the extreme values of negative log-likelihood that can be potentially caused by small variance. We believe that the v-noise helps prevent the network from being biased and it is better than using constant regularization as this v-noise depends on particular dataset by using the standard deviation of the training data multiplied by a manually tuned scalar factor.

3.1.2 Reproduction of Deep Ensemble

We reproduce Deep Ensemble using a similar experimental setup as we did for MC-Dropout. However, some differences exist. Firstly, the network does not use dropout. Instead, 10 separate networks are created for training on each dataset. For each created network, while remaining the same architecture and loss function of the network, the weights are randomly initialized at the beginning. During evaluation, the input is fed into all trained networks to generate the sample. The mean and standard deviation are used for network prediction and model uncertainty similar to MC-Dropout and are evaluated using the same mechanism as MC-Dropout.

3.1.3 Reproduction of Parametric Uncertainty

For MC-Dropout and Deep Ensemble models, MSE was selected as a training criterion in regression tasks and showed good performance with v-noise. However, the MSE is not capable of capturing the uncertainty. The necessity of directly capturing the uncertainty in regression tasks led to the introduction of new criterion: the negative log-likelihood (NLL) under an assumption that observed values are sampled from the Gaussian distribution [5].

$$-\log p_\theta(y_n|x_n) = \frac{\log \sigma_\theta^2(x)}{2} + \frac{(y-\mu_\theta(x))^2}{2\sigma_\theta^2(x)} + constant \tag{4}$$

In order to apply this criterion as a loss function of model, we adjust the last layer of the network, which has the same structure as the one in Deep ensemble method, to outputs two predictions(hidden units) $\hat{y_1}$ and $\hat{y_2}$ corresponding to the predicted mean $\mu(x)$ and variance $\sigma^2(x)$, respectively. Note that we applied the natural exponential function to $\hat{y_2}$, meaning predictive variance $\sigma^2(x) = \exp(\hat{y_2})$, in order to ensure predicted $\sigma^2(x)$ has a positive value. As now the model directly predicts the variance σ^2, we no longer need a multiple forward pass to compute the predictive variance as we did in 3.1.1 and 3.1.2.

While the general network structure remains the same as Deep Ensemble models apart from the last layer, the choice of some hyperparameters has been slightly changed. Firstly, it has been observed that the learning rate of 0.1 is too large and even make some datasets non-trainable. We figured out that applying a smaller learning rate (e.g. 0.001) can resolve this issue. Here, one can consider further investigation: applying a learning rate scheduler in case a lower learning rate does not solve the problem. Secondly, the number of epoch changed from 40 to 150. We found the network can be further trained over epoch 40 and showed better performance. Lastly, in order to prevent NLL loss from being minus infinity in training, we cap the computed loss at -100.

3.1.4 Experiment results on UCI Datasets

This section shows the results we obtained from the three methods. We compare our results with the ones reported in [5]. One can observe that most of the datasets have achieved comparable results. Note that results of MC-Dropout and Deep Ensembles are from models trained for 40 epochs with learning rate 0.1 while results of Parametirc Uncertainty are from models trained for 150 epochs with learning rate 0.001

Dataset	RMSE			NLL		
	MC-Dropout	Deep Ensembles	Parametric	MC-Dropout	Deep Ensembles	Parametric
boston	3.32	3.17	2.78	4.22	5.37	2.20
concrete	5.59	6.09	5.31	2.93	2.94	2.97
energy	2.29	2.08	2.28	3.14	3.54	1.41
kin8nm	0.10	0.08	0.07	-1.48	-1.82	-1.93
naval	0.09	0.06	0.07	1.32	0.04	5.79
power_plant	3.51	3.12	3.05	2.88	2.86	2.89
protein	4.29	3.58	3.47	2.19	2.09	2.19
wine	0.51	0.49	0.50	1.08	1.63	0.02
yacht	4.93	5.87	2.37	3.69	18.15	0.94

Table 1: Results of the 3 methods applied to the UCI datasets used in [5]

3.2 Uncertainty realism

As uncertainty mechanisms measure how much our model is unsure of its prediction, it needs to be realistic. In other words, measured uncertainty should ideally grow as the residual error $y - \mu(x)$

increases, and vice versa. In order to assess the realism of uncertainty mechanisms, many researches have been performed to find appropriate metrics [6, 7].

3.2.1 Uncertainty realism assessment on MC-Dropout

Here, in order to assess the realism of MC-Dropout method over UCI datasets, we used the proposed criterion for 1D regression task with Mahalanobis distances M in [7]:

$$M^2_{\mu_i,\sigma_i}(y_i) = \frac{\xi_i^2}{\sigma_i^2} = \frac{(y_i - \mu_i)^2}{\sigma_i^2} \sim \chi^2(d=1) \tag{5}$$

where $y_{i,gt}$ is target, $y_{i,sample}$ is model output from x_i, (μ_i, σ_i) is predicted mean and standard deviation given the set of generated samples $\{\hat{y}^k_{i,sample}\}$ from model and ξ_i denotes a residual error $y_i - \mu_i$. As we assume observed samples $y_{i,sample}$ follow a Gaussian distribution, $\frac{\xi_i}{\sigma_i}$ is a random variable from standard normal distribution and therefore, the set $M_{sample} = \{M^2_{\mu_i,\sigma_i}(y_{i,sample})\}$ follows a chi-square distribution $\chi^2(d = 1)$. Further, since an observation $y_{i,gt}$ is also assumed to be an realization of a Gaussian distribution. This means that if the network estimates, μ_i and σ_i, are realistically approximated to true Gaussian parameters then M_{gt} should also follows $\chi^2(d = 1)$

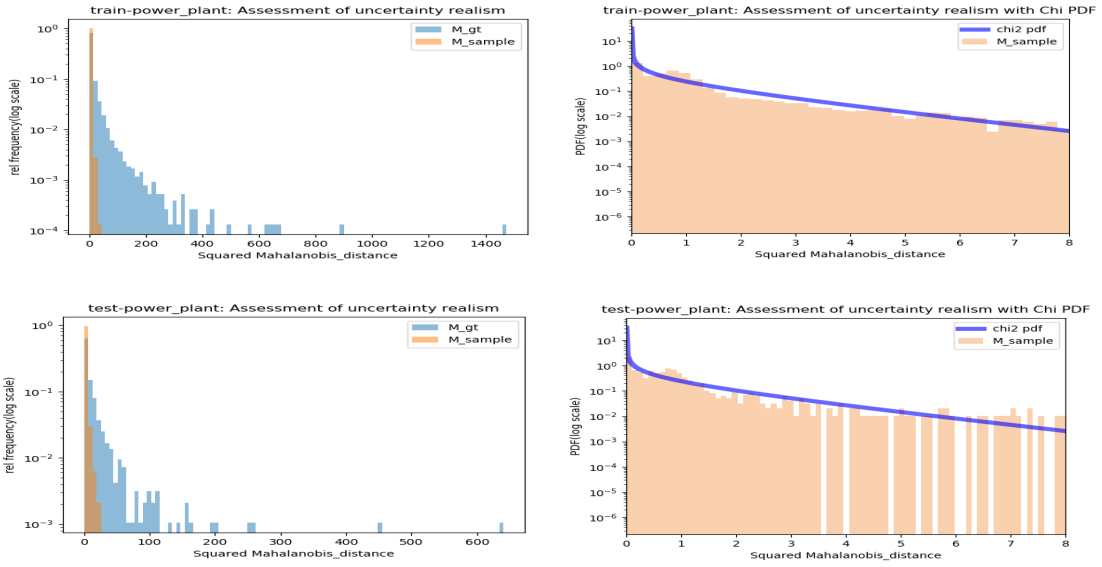
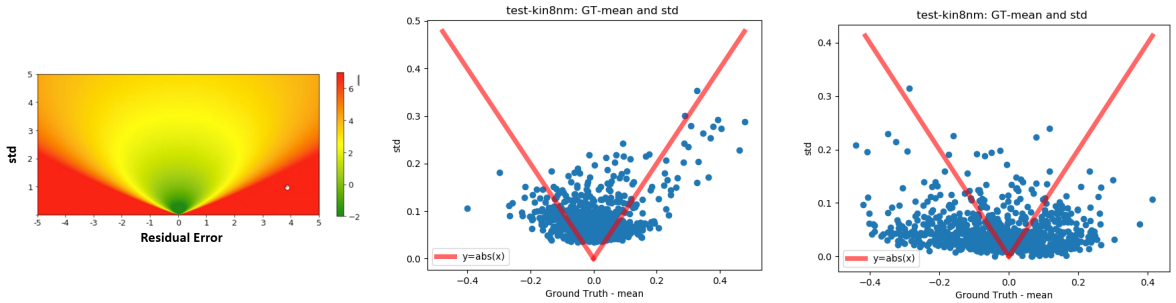


Figure 1: Uncertainty realism assessment for 1D regression task given train and test set of Power Plant dataset. Left: empirical distributions of M_{gt} and M_{sample}, right: distribution of M_{sample} and PDF of $\chi^2(d = 1)$ with constrained x-axis $\in [0, 8]$, upper row: evaluation on train set, lower row: evaluation on test set.

However, as is shown in figure 1 and in [7], two distributions show clearly different shapes. While M_{sample} has a shape of probability density function (PDF) of $\chi^2(d = 1)$, the PDF of M_{gt} largely deviates from M_{sample}. Based on this observation, we have decided that the MC Dropout method does not produce realistic estimates in our experiments.

3.2.2 Comparison of MC-Dropout and PU

As is shown in the left panel of figure 2, NLL has a strong gradient around $y = abs(x)$, and its value decreases as (ξ, σ) goes to a region above the V-shape line and to the center where $\xi = 0$. This



(a) Heatmap of NLL (b) Scatter plot from PU (c) scatter plot from MC-Dropout

Figure 2: Evaluation of Uncertainty realism with PU and MC-Dropout on kin8nm dataset. Left: Heatmap of NLL, middle and right: predictive standard deviation w.r.t residual error by PU and MC-Dropout, respectively

characteristic of NLL enforces the PU to learn an ideal correlation between ξ and σ: as an absolute of residual error grows a predicted standard deviation also increases.

This, in our experiments, leads PU (Figure 2, middle panel) to produce more realistic uncertainties compared to MC-Dropout (Figure 2, right panel) as NLL is used as a loss function in PU. Plotted points indicate $\{\xi_i, \sigma_i\}$ evaluated by respective uncertainty mechanism given test set.

4 Study on Out of Distribution Data

In many areas, especially safety-oriented area such as autonomous driving, it is important to analyze the ability of a model to generalize its performance to data that has not been seen in the training examples. It is not surprising that a model misbehave given unseen data. However, a model which is capable of modeling uncertainty is expected to give higher uncertainty to unseen data. This makes sense as the data is unfamiliar to the model.

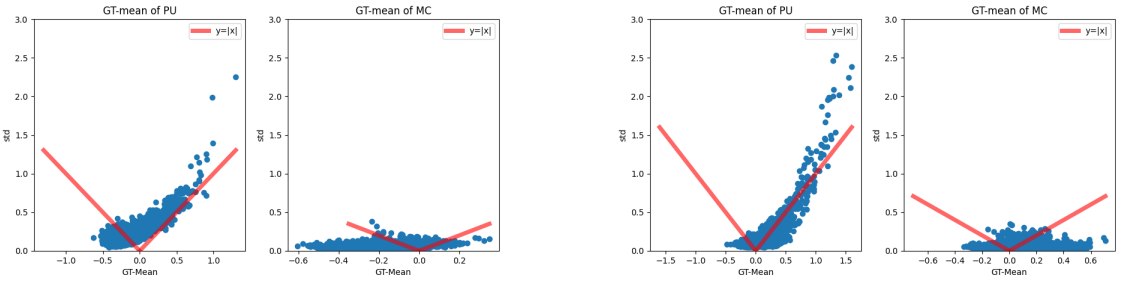
In this section, we illustrate our study of MC-Dropout and Parametric Uncertainty being applied to Out of Distribution Data (OOD data). Deep Ensemble is not considered here as it has the mechanism of uncertainty estimation similar to MC-Dropout and one can expect similar performance compared with MC-Dropout. We demonstrate the production of the OOD data. Subsequently, we compare and analyze the results we obtain from the experiments.

4.1 Production of OOD dataset

We use the currently studied datasets for generating the OOD data by firstly ordering the data points by target values. After the dataset has been ordered, a split from the middle separates the dataset into 50% data with smallest target values and another 50% data with largest targets. We use the data with 50% small targets as training data and the other part as testing data. By doing this, we assume that the distribution of the testing data is to some extent far away from the training data. Although it is more direct to split the data based on input features and the generated data is intuitively far away from each other, we argue that splitting on target values is still a reasonable yet much simpler way to generate OOD data. Data with larger labels, compared with data having small labels, already indicates potential difference in input feature space. Therefore, we believe that splitting data on targets is already enough for simple datasets used in the experiment while still retaining simplicity.

4.2 Experiment Result

We focus on comparing MC-Dropout and Parametric Uncertainty because one of the key difference between these two methods is that the MC-Dropout models uncertainty using standard deviation of the sample of network outputs (similar principles apply to Deep Ensemble) while Parametric Uncertainty directly learns to model the uncertainty from data. To have a better comparison, we use a similar experimental setup as we did for normal datasets with several differences. We set the learning rate of MC-Dropout to 0.001 and train the network for 150 epochs which is also used by Parametric Uncertainty. We also experimented with training MC-Dropout for 40 epochs and obtained similar results. Therefore, we choose 150 epochs for MC-Dropout to keep the setup as consistent as possible.



(a) kin8nm train dataset

(b) kin8nm test dataset

Figure 3: Scatter plots of standard deviation with respect to residual from Parametric Uncertainty and MC-Dropout applied to kin8nm dataset. We choose kin8nm as it represents the most datasets where Parametric Uncertainty outperforms MC-Dropout

Dataset	RMSE		NLL		NLL without v-noise	
	Parametirc	MC-Dropout	Parametirc	MC-Dropout	Parametirc	MC-Dropout
boston	12.85	12.85	57.48	263.13	64.94	655.46
concrete	27.82	26.96	7.69	8.23	39.98	254.27
energy	18.08	18.09	365.37	1235.72	612.41	2710.22
kin8nm	0.36	0.37	8.63	25.16	18.53	180.35
naval	0.03	0.03	0.23	1.05	42.89	154.13
power_plant	28.77	28.70	10.21	12.02	58.92	1402.25
protein	10.45	10.48	32.38	49.96	121.19	4259.02
wine	1.30	1.31	13.18	43.01	27353421.38	502.22
yacht	18.94	18.88	4421.82	15513.63	4608.79	16343.95

Table 2: Results of the 3 methods applied to the OOD data. The data is generated from the studied UCI datasets that are split based on the target features.

From the results, one can observe that the networks, in general, cannot give the best performance due to unseen data. However, we found that Parametric Uncertainty tends to give slightly better behavior on some datasets (e.g. kin8nm as shown in Figure 3) compared with MC-Dropout. One possible reason is that Parametric Uncertainty learns to model uncertainty from data which gives it better ability to describe unknown knowledge. Further, previous the study has shown the limitation of MC-Dropout modeling uncertainty for network outputs being far away from ground truth. However, one could also expect that MC-Dropout would outperform PU on data that is even further away from the training data. This is reasonable because MC-Dropout is an ensemble method where individual networks contribute to the output. For OOD data it is more likely to give higher uncertainty since the individual networks will produce rather uneven estimates. The train and test data are not far away enough from each other in our experiment and therefore, might not be the best use case for MC-Dropout. A more aggressive splitting mechanism can be used to study the methods further, and we leave this to future work.

5 Discussion

We have reproduced comparable results of MC-Dropout, Deep Ensemble and Parametric Uncertainty on UCI regression datasets having 1D target features. In general, our experiment suggests that Parametric Uncertainty outperforms the other two methods as it reports lower NLL values with and without v-noise, indicating that Parametric Uncertainty is more stable, and produces more realistic uncertainty estimates. Furthermore, Parametric Uncertainty shows its strength over MC-Dropout in the study of model performance on OOD data produced from some of the datasets in our experiment. Meanwhile, we also find that both methods can collapse for other datasets. We argue that Parametric Uncertainty yields better performance because the Parametric Uncertainty method learns the uncertainty directly from data, whereas MC-Dropout produces its estimates based on sample statistics. What's more, both methods proved to be affected by the distribution of input data and it is possible that MC-Dropout will become better due to its nature if the produced OOD data is further distributed from training data.

While Parametric Uncertainty outperforms MC-Dropout and Deep Ensemble on regression tasks, it is not as easy as the other two methods to be applied on classification tasks. Meanwhile, MC-Dropout and Deep Ensemble have the simplicity and flexibility in practice but they produce less realistic uncertainty estimation. One good future direction would be combining those methods to discover possibility of better approaches of uncertainty estimation.

For another further investigation, we found that MC-Dropout with UCI datasets is sensitive to the learning rate. As is shown in table 4, reported NLL of MC-Dropout model trained with the learning rate 0.1 for some datasets were unrealistically huge. We resolved this issue by introducing the v-noise to prevent extreme NLL. While conducting experiments with different choices for hyperparameters, we figured out applying smaller learning rate(e.g. 0.001) can produce stable and reasonably small NLL values even without the existence of v-noise as is reported in table 3.

The study of application of Parametric Uncertainty in classification tasks and sensitivity of MC-Dropout to a learning rate can be included in future work.

References

[1] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[2] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[3] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

[4] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105 – 112, 2009. Risk Acceptance and Risk Communication.

[5] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[6] Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *CoRR*, abs/1811.12709, 2018.

[7] Sicking, Kister, and Fahrland et al. Approaching neural network uncertainty realism. https://ml4ad.github.io/files/papers/Approaching2019.

Appendices

A Additional results

Datasets	RMSE	NLL	NLL_no_v_noise
boston	3.24	3.80	6.61
concrete	6.08	2.95	6.19
energy	2.31	3.01	5.12
kin8nm	0.10	-1.50	1.06
naval	0.04	-0.87	9.53
power_plant	3.20	2.87	7.23
protein	3.62	2.10	35.69
wine	0.50	1.56	33.51
yacht	6.10	11.76	13.11

Table 3: Results of MC-Dropout trained for 40 epochs with learning rate 0.001

Dataset	RMSE			NLL			NLL without v-noise		
	MC-dropout	Deep Ensembles	Parametric	MC-dropout	Deep Ensembles	Parametric	MC-dropout	Deep Ensembles	Parametric
boston	3.32	3.17	2.78	4.22	5.37	2.2	6.61	29.83	2.70
concrete	5.59	6.09	5.31	2.93	2.94	2.97	6.19	19.75	2.87
energy	2.29	2.08	2.28	3.14	3.54	1.41	5.12	15.81	10.97
kin8nm	0.1	0.08	0.07	-1.48	-1.82	-1.93	1.06	4.63	-2.01
naval	0.09	0.06	0.07	1.32	0.04	5.79	9.53	2.20	218.03
power_plant	3.51	3.12	3.05	2.88	2.86	2.89	7.23	84.19	1.86
protein	4.29	3.58	3.47	2.19	2.09	2.19	35.69	43.67	1.89
wine	0.51	0.49	0.5	1.08	1.63	0.02	33.51	117.09	0.64
yacht	4.93	5.87	2.37	3.69	18.15	0.94	13.11	23.32	0.97

Table 4: Complete results of the 3 methods on normal datasets, including results without applying regularization 'v-noise'