On the Validity of Bayesian Neural Networks for Uncertainty Estimation

John Mitros and Brian Mac Namee

School of Computer Science
University College Dublin, Dublin, IR
{ioannis, brian.macnamee}@insight-centre.org

Abstract. Deep neural networks (DNN) are versatile parametric models utilised successfully in a diverse number of tasks and domains. However, they have limitations—particularly from their lack of robustness and over-sensitivity to out of distribution samples. Bayesian Neural Networks, due to their formulation under the Bayesian framework, provide a principled approach to building neural networks that address these limitations. This work provides an empirical study evaluating and comparing Bayesian Neural Networks to their equivalent point estimate Deep Neural Networks to quantify the predictive uncertainty induced by their parameters, as well as their performance in view of uncertainty. Specifically, we evaluated and compared three point estimate deep neural networks against their alternative comparable Bayesian neural network utilising well-known benchmark image classification datasets.

Keywords: Bayesian Neural Networks, Uncertainty Quantification, OoD, Robustness

1 Introduction

With the advancement of technology and the abundance of data, our society has been transformed beyond recognition. From smart home assistance technologies to self-driving cars, to smart mobile phones a multitude of connected devices now assist us in our daily routines.

One thing that is common among these devices is the exponential explosion of data generated as a consequence of our activities. Predictive models rely on this data to capture patterns in our daily routines, from which they can offer us assistance tailored to our individual needs. Many of these predictive models are based on deep neural networks (DNNs).

The machine learning community, however, is becoming increasingly aware of issues associated with DNNs ranging from fairness to bias, and, from robustness to uncertainty estimation. Motivated by this we setup to investigate the issues of reliability and trustworthiness of prediction confidence estimates produced by DNNs. We first assess the capability of current DNN models to provide confident (i.e. calibration error) and reliable (i.e. noise sensitivity error or the ability to predict out of sample instances with high uncertainty) predictions. Second, we compare this to the capability of equivalent recent Bayesian formulations (i.e. Bayesian neural networks (BNN)), in terms of accuracy, calibration error, and ability to recognise and indicate out of sample instances.

There exist two types of uncertainties related to predictive models, *aleatoric* and *epistemic* uncertainty [4]. Aleatoric uncertainty is usually attributed to stochasticity inherent in the related task or experiment to be performed. It can therefore be considered an irreducible error. Epistemic uncertainty is usually attributed to uncertainty induced by the model parameters. It can therefore be considered a a reducible error as it can be reduced by obtaining more data. The question under investigation in this work is whether BNNs can provide better calibrated and reliable estimates for out of sample instances compared to point estimate DNNs and therefore relates to epistemic uncertainty.

The remaining sections of this paper are divided as follows. Section 2 outlines related work in the area of confident estimations and Bayesian neural networks. In Section 3, we provide the information related to the datasets used throughout the experiments, their respective sizes and types. In Section 4, we describe the metrics used to evaluate whether a classifier is calibrated (i.e. expected calibration error and reliability figures), as well as its ability to identify and predict out of sample instances (i.e. symmetric KL divergence and distributional entropy figures), along with their respective explanations. In Section 5, we introduce the three BNN approaches utilised in the experiments, providing detailed explanations of how they work. Finally, in Sections 6 and 7 we present the results related to confidence calibration (i.e. Table 1 and Figure 1) and reliability prediction estimates for out of sample instances (i.e. Table 3 and Figures 2), along with the concluding remarks.

2 Related Work

Earlier findings [13, 17, 20, 18, 1] have demonstrated the incapacity of point estimate deep neural networks (DNN) to provide confident and calibrated [10, 5, 9] uncertainty estimates in their predictions. Motivated by recent work in this area we strive to demonstrate, first, that this is indeed a serious problem currently investigated in the machine learning community, and second, provide an alternative viable solution (i.e. BNN) which combines the best from both worlds (i.e. a principled and elegant formulation due to the Bayesian inference framework and powerful and expressive models thanks to DNN).

In order to help the reader understand the terminology and semantics of uncertainty quantification in predictive models, it would be helpful to relate the variance existent in the model parameters represented as the sum of both aleatoric and epistemic uncertainty. Additionally, whenever the reader encounters the following terminology "point estimate DNN" in the document, it simply refers to a DNN model coupled with a softmax function in the final layer. For instance, suppose our DNN is defined by $\hat{\mathbf{y}} = f(\mathbf{x})$, where $\hat{\mathbf{y}}$ denotes the predictions for K possible classes. Then a "point estimate DNN" is simply defined by exponentiating each prediction and normalising it by the total sum of all exponentiated predictions.

$$p(y = j|\mathbf{x}) = \frac{e^{\hat{\mathbf{y}}_i}}{\sum_{j=1}^{K} e^{\hat{\mathbf{y}}_j}} \text{ for } i = 1, \ldots, K \text{ and } \hat{\mathbf{y}} = (y_1, \ldots, y_K) \in \mathbb{R}^K \qquad (1)$$

The reason for identifying them as point estimate DNN is because they are mistakenly misinterpreted as probabilistic models due to the fact that they provide predictions which resemble probabilities (i.e. estimates $\in \mathbb{R}_{[0-1]}$). Furthermore, Eq. 1 is misinterpreted mistakenly for a categorical distribution to which we disagree since it should have a prior Dirichlet in order to be classified as a categorical distribution. Our view is that Eq. 1 is more of a mathematical convenience in order to allow DNN model to emit predictions rather than a well defined probability distribution.

As previously stated there are two main problems investigated in this work. The first one is related to the inability of DNN to predict probability estimates representative of the true correct likelihood function (i.e. calibration confidence). For instance, in a binary classification task a classifier is considered uncalibrated when the classifiers' predictions do not match the empirical proportion of the positive class upon which the classifier is requested to make a prediction. Poor calibration confidence problems in DNN can be affected by different choices while constructing the DNN architecture [5, 9] (e.g. depth, width, regularisation or batch-normalisation). The second problem, is related to the incapacity of DNN to identify and reliably predict out of sample instances (i.e. noise sensitivity) which can be a consequent of noise in the data, noise in the model parameters or noise constructed by an adversary in order to manipulate the models' predictions [1, 11, 18, 3].

3 Data

The data used in this empirical study include two well-established datasets in the machine learning literature, *CIFAR-10* [8] and *SVHN* [14]. Both datasets are comprised of colour images of dimensionality 32x32 and include 10 distinct categories. In addition, both are considered to represent real world datasets with *CIFAR-10* being collected over the Internet while *SVHN* [14] being a result of the Google Street View project representing house numbers. Further details regarding the number of instances of each dataset and equivalently their categories are described below

– The *CIFAR-10* [8] dataset consists of 60,000 colour images of dimensionality 32x32 with 10 classes. Each class contains 6,000 images. In total there are 50,000 training images and 10,000 test images.
– The *SVHN* [14] dataset consists of 99,289 colour images of dimensionality 32x32 representing digits of house numbers. There exist 10 categories one for each digit, in total there are 73,257 colour images representing digits for training, and equivalently 26,032 digits for testing.

4 Metrics

The chosen evaluation metrics utilised for this empirical study involved:

– Accuracy
– Expected calibration error
– Entropy
– Symmetric D_{KL} divergence

Particularly, for a given neural network model $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$ of depth L defined as $\{\mathbf{W}_L \sigma_L(\mathbf{W}_{L-1} \ldots \sigma_2(\mathbf{W}_2 \sigma_1(\mathbf{W}_1 \mathbf{x})))\}$, describing the number of function compositions and parameters $\{\boldsymbol{\theta} = \{\mathbf{W}_1, \ldots, \mathbf{W}_L\}$ with $\sigma(\cdot)$ being a non-linear function.

The accuracy on a given output $\hat{y}_n$ is measured by the indicator function $\mathrm{acc} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}(y_n \neq \hat{y}_n)$ for each instance $n \in N$ averaged over the total number of instances N in the dataset. This metric is predominantly used in the machine learning community to evaluate the generalisation ability of a predictive model on a hold-out test set.

In order to capture whether a model is calibrated we utilised the expected calibration error $ECE = \sum_{m=1}^{M} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \frac{|B_m|}{N}$ in combination with the equivalent reliability plots shown in Figure 1 similar to [5]. ECE is usually expressed as a weighted average between the accuracy and confidence of a model across M bins for N samples. This metric has the ability to capture any disagreement between the classifiers predictions and the true empirical proportion of instances for each class category for every mini-batch of instances presented to the classifier.

Furthermore, to assess a models' ability to characterise out of sample data with high degree of uncertainty we focused on the work of [11] utilising information entropy $H(Y) = -\sum_{k=1}^{K} p(\hat{\mathbf{y}}_k) \log p(\hat{\mathbf{y}}_k)$ on the final predictions of a model in order to derive the uncertainty plots depicted in Figure 2. Essentially, for every input $\mathbf{x}$ we have we have a corresponding vector of predictions $\hat{\mathbf{y}} = (0.86, 0.23, \ldots, K)$ where each entry denotes the prediction of the classifier for each class $k \in K$. For every dataset we split them randomly into two halves. The first half represents the $K/2$ classes and the other other half the remaining. We select one of the halves to be utilised to train the classifier (i.e. denotes in-sample instances) and the remaining half (i.e. denotes out-of-sample instances) to be utilised only during the testing phase of the classifier. Therefore, after the classifier has been trained on one half (hence the 5+5 categories in Figures 2) we evaluate its generalisation ability on the remaining half where for every input we have a corresponding entropy over the K classes. This provides a distribution over the total number of N inputs allowing to distinguish and evaluate the classifier entropy among in-sample vs out-of-sample instances.

Finally, in order to conveniently compare and summarize a models' performance on detecting out of sample instances as a summary statistic the scalar value of the symmetric KL-divergence $D_{KL}(p \parallel q) + D_{KL}(q \parallel p)$ [11] between two distributions p and q was selected as a sensible candidate. The choice of

KL-divergence allows to evaluate how similar are two distributions p and q. The larger the KL-divergence is the more distinct are the distributions p and q. Since we want to evaluate the ability of the classifier to recognise out of sample instances we should be able to measure the KL-divergence of the classifiers' estimates for in-sample p against out of sample q instances. The larger KL values denote the classifier is in better position to recognise out of sample instances.

5 Methods

This section provides the details of the empirical evaluation comprised of the following three components, (i) models, (ii) calibration and (iii) uncertainty. The following (i) models were selected among which three of them represent point estimate deep neural networks (DNN) and the remaining three their equivalent Bayesian Neural Networks (BNN).

- Point estimate deep neural networks
  - VGG16 [19]
  - PreResNet164 [6]
  - WideResnet28x10 [22]
- Bayesian neural networks
  - VGG16 - Monte Carlo Dropout [4]
  - VGG16 - Stochastic Weight Aaveraging of Gaussian samples [11]
  - Stochastic Variational - Deep Kernel Learning [21]

(ii) The calibration of each model was evaluated using the expected calibration error introduced in Section 4 in combination with the reliability plots demonstrated in Figure 1. Each model was trained on 5 categories from *CIFAR-10* and accordingly *SVHN* with the remaining 5 categories being whithheld in order to evaluate the models' ability to associate out of samples instances with high uncertainty as they were not introduced to the model at any step. The duration of training for each model was 300 epochs with best performing model on the validation set being selected as the final model for each architecture.

(iii) As already stated in order to evaluate a model's ability to detect out of sample instances with high uncertainty we utilised entropy on the predictions of a model to derive Figures 2, for each dataset and model combination accordingly. In addition, the symmetric KL-divergence described in Section 4 was introduced in order to provide a comparable scalar summary statistic of the overall essence of Figures 2.

In the remainder of this section we will introduce the three Bayesian neural network approaches utilised during the experimental study:

1. Dropout as Bayesian approximation:
   Provides a view of dropout at test time as approximate Bayesian inference [4]. It is based on prior work of [2] which established a relationship between

neural networks with dropout and Gaussian Processes (GP). Given a dataset $(\mathbf{X},\mathbf{Y})$ the posterior over the GP is formulated as,

$$\begin{aligned}
&\mathbf{F}|\mathbf{X} \sim \mathcal{N}(0,\mathbf{K}(\mathbf{X},\mathbf{X}))\\
&\mathbf{Y}|\mathbf{F} \sim \mathcal{N}(\mathbf{F},0\cdot\mathbf{I})\\
&\hat{y}|\mathbf{Y} \sim \text{Categorical}\,(\cdot)
\end{aligned}$$

where $\hat{y}$ denotes a class label and $\hat{y} \neq \hat{y}\prime$. An integral part of the GP is the choice of the covariance matrix $\mathbf{K}$ representing the similarity between two inputs as a scalar value. The key insight to draw connections between neural networks and Gaussian processes is to consider the possibility the choice of the kernel to represent a non-linear function, for instance, consider the rectified linear (ReLU) function, then the kernel would be expressed as $\int p(\mathbf{w})\sigma(\mathbf{w}^T\mathbf{x})\sigma(\mathbf{w}^T\mathbf{x})d\mathbf{w}$ with $p(\mathbf{w}) \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$. Because usually the integral is intractable a conventional approach would be to use Monte Carlo integration in order to approximate it $\hat{k} = \frac{1}{T}\sum_{t=1}^{T}\sigma(\mathbf{w}_t^T\mathbf{x})\sigma(\mathbf{w}_t^T\mathbf{x})$, hence, the name Monte Carlo Dropout. Let us now consider a one hidden layer neural network with dropout $\hat{y} = (\beta_2\mathbf{W}_2)\sigma(\mathbf{x}(\beta_1\mathbf{W}_1))$ where $\beta_1,\beta_2 \sim \text{Bernoulli}(p_{1,2})$. Utilising the approximate kernel $\hat{k}$ one can express the parameters $\mathbf{W}_{1,2}$ as $\mathbf{W}_{1,2} = \beta_{1,2}(\mathbf{A}_{1,2}+\boldsymbol{\sigma\epsilon}_{1,2})(1-\beta_{1,2})\boldsymbol{\sigma\epsilon}_{1,2}$ with $\mathbf{A}_{1,2},\boldsymbol{\epsilon}_{1,2} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ and $\beta_{1,2} \sim \text{Bernouli}(p_{1,2})$ closely resembling the NN formulation. Therefore, to establish the final connection among NNs trained with stochastic gradient descent (SGD) and dropout to GPs one has to simulate Monte Carlo sampling by drawing samples from the trained model at test time $[\hat{y}_n = f(\mathbf{x}_n;\beta_n\boldsymbol{\theta}_n)]_{n=1}^{N}$ with $\beta_n \sim \text{Bernouli}(p_n)$. The samples $\hat{y}_n$ resulting from the different dropout masks β_n are averaged over the N different models in order to approximate and retrieve the posterior distribution.

2. Stochastic weight averaging of Gaussian samples
Stochastic weight averaging of Gaussian samples (SWAG) [11] is an extension of stochastic weight averaging (SWA) [7] where the weights of a NN are averaged during different SGD iterates, which in itself can be viewed as approximate Bayesian inference [12], with ideas traced back to [16, 15]. In order to understand SWAG we first need to explain SWA. SWA at a high level can be viewed as averaged SGD [16, 15]. Essentially the main difference between SWA and averaged SGD is that SWA utilises a simple moving average instead of an exponential one, in conjunction with a high constant learning rate, instead of a decaying one. In essence, in SWA one maintains a running average over the weights of a NN during the last 25% of the training process which is used to update the first and second moments of batch-normalisation. This leads to better generalisation since the SGD projections are smoothed out during the average process leading to wider optima in the optimisation landscape of the NN. Now that we have established what SWA is let us introduce SWAG. SWAG is an approximate Bayesian inference technique for estimating the covariance from the weight parameters of a NN. SWAG maintains a running average $\bar{\theta^2} = \frac{1}{T}\sum_{t=1}^{T}\theta_t^2$ in order to compute the covari-

ance $\boldsymbol{\Sigma} = \text{diag}(\bar{\theta^2} - \theta^2)$ which produces the approximate Gaussian posterior $\mathcal{N}(\theta, \boldsymbol{\Sigma})$. At test time the weights of the NN are drawn from this posterior $\tilde{\theta_n} \sim \mathcal{N}(\theta, \boldsymbol{\Sigma})$ in order to perform Bayesian model averaging to retrieve the final posterior of the model as well as the uncertainty estimates from the first and second moments.

3. Deep kernel learning
   The deep kernel learning method [21] establishes a combination of NN architectures and GPs trained jointly in order to derive kernels with GP properties overcoming the need to perform approximate Bayesian inference. The first part is composed of any NN (i.e. task dependent) whose output is utilised in the second part in order to approximate the covariance of the GP in the additive layer As explained earlier in the MC-Dropout approach a kernel between inputs $\mathbf{x}$ and $\mathbf{x\prime}$ can be expressed via a non-linear mapping function thanks to the kernel trick $k(\mathbf{x}, \mathbf{x\prime}) \to k(f(\mathbf{x}, \mathbf{w}), f(\mathbf{x\prime}, \mathbf{w})|\mathbf{w})$ therefore combining NNs with GPs seems like a natural evolution which permits scalable and flexible kernels represented as neural networks to be utilised directly in Gaussian Processes. Finally, given that the formulation of GPs allows to represent a distribution over a function space it is thus possible to derive uncertainty estimates from the moments of this distribution in order to inform the models about the uncertainty in their parameters having an impact in the final posterior distribution.

6 Results

In this section we describe the results from our experiments and the findings that arise from these for our two initial questions. Let us recall them here again for clarity:

– Do point estimate deep neural networks suffer from pathologies of poor calibration and inability to identify out of sample instances?
– Are Bayesian neural networks better calibrated and more resilient to out of sample instances?

To answer the first question we draw the attention of the reader to Figure 1 and equivalently Table 1. Figure 1 shows the reliability plots for all models and datasets. In these plots a perfectly calibrated model is indicated by the diagonal line. Anything below the diagonal represents an over-confident model, while anything above the diagonal represents an under-confident model. The expected calibration errors (ECE) in Table 1 (which measure the degree of miscalibration present) seem to be in accordance with the results from [5]. All of the models are somewhat miscalibrated. Some of the Bayesian approaches, however—in particular the models based on MC-Dropout and SWAG—are better calibrated than their point estimate DNN counterparts.

Notice that all models exhibit high accuracy on the final test set (shown in Table 2). This illustrates that a model can be very accurate but miscalibrated,

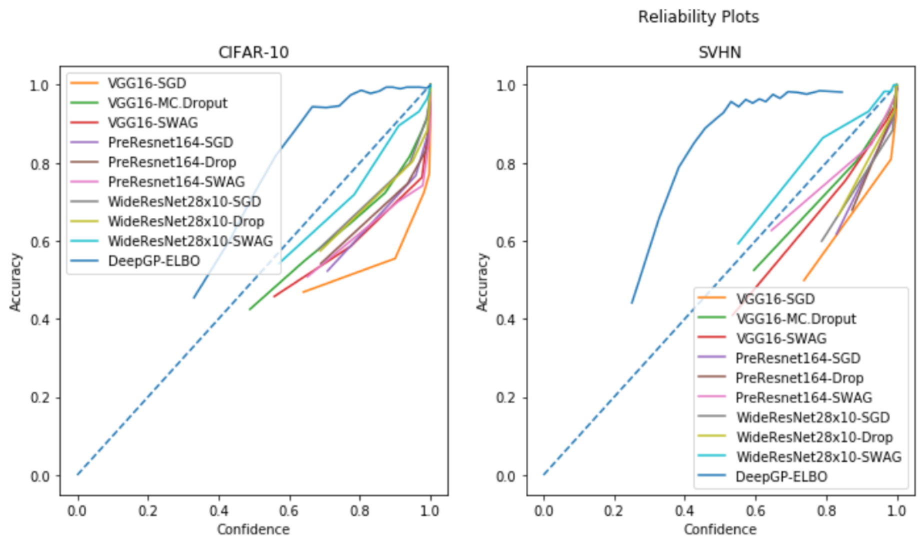

Fig. 1: Reliability plots across all models on *CIFAR-10* [8] and *SVHN* [14] datasets.

Table 1: Expected calibration errors (ECE) for *CIFAR-10* and *SVHN* equivalently. Lower values indicate better calibrated models.

Models	CIFAR10	SVHN
VGG16-SGD	0.0677236	0.0307552
VGG16-MC Dropout	0.0423307	0.0155248
VGG16-SWAG	0.0499478	0.0204564
PreResNet164	0.0309783	0.0238022
PreResNet164-MC Dropout	0.0338277	0.0161251
PreResNet164-SWAG	0.049338	0.0089618
WideResNet28x10	0.0200257	0.0210420
WideResNet28x10-MC Dropout	0.0255283	0.0147181
WideResNet28x10-Swag	0.0097967	0.0082371
DeepGaussProcess	0.1418236	0.3275412

or equivalently a model can be very well calibrated but inaccurate. There is no real correlation between between calibration and accuracy of a model. It is also known [5] that as the complexity of the model increases the calibration error increases as well.

In order to evaluate and demonstrate the ability of the models to handle out of sample instances we divided each of the *CIFAR-10* and *SVHN* datasets into two halves containing 5 categories each. These partitions represent in and out of distribution samples. In the discussion that follows this is indicated with the parenthesis (5 + 5) next to the dataset name to denote that the model was trained on only 5 categories representing in distribution samples and at test time

Table 2: Accuracy of all the models on both datasets *CIFAR-10* and *SVHN*.

Models	CIFAR10	SVHN
VGG16-SGD	94.40	97.10
VGG16-MC Dropout	93.26	96.87
VGG16-SWAG	93.80	96.83
PreResNet164	93.56	97.90
PreResNet164-MC Dropout	94.68	97.73
PreResNet164-SWAG	93.14	97.69
WideResNet28x10	94.04	97.44
WideResNet28x10-MC Dropout	95.54	97.63
WideResNet28x10-SWAG	95.12	97.95
DeepGaussProcess	91.00	93.00

it was evaluated on the other 5 categories to simulate out of sample instances. The results are illustrated in Figures 2, for *CIFAR-10* and *SVHN* respectively, and summarised in Table 3. The information depicted in Table 3 provides a summary of Figures 2, by measuring the symmetric KL divergence, between the distribution of class confidence entropies of each model for the in and out of sample instances.

Table 3: Symmetric D_{KL} divergence between in and out of distribution splits of *CIFAR-10* (5 + 5) and *SVHN* (5 + 5). Higher values indicate the ability of a model to flag out of sample instances with high uncertainty.

Models	CIFAR10	SVHN
VGG16-SGD	2.952740	5.638210
VGG16-MC Dropout	3.849440	6.273212
VGG16-SWAG	2.375000	5.058158
PreResNet164	4.244287	3.375254
PreResNet164-MC Dropout	2.879347	2.705263
PreResNet164-SWAG	1.810131	3.344560
WideResNet28x10	2.181033	3.051318
WideResNet28x10-MC Dropout	2.929135	2.995543
WideResNet28x10-SWAG	2.780160	3.646878
DeepGaussProcess	0.801246	0.153648

Together these results suggest that the Bayesian methods are better at identifying out of sample instances. Although the result is not clear cut, in some cases the point estimate networks get higher divergence scores than the Bayesian ones, overall the results point in the Bayesian direction.

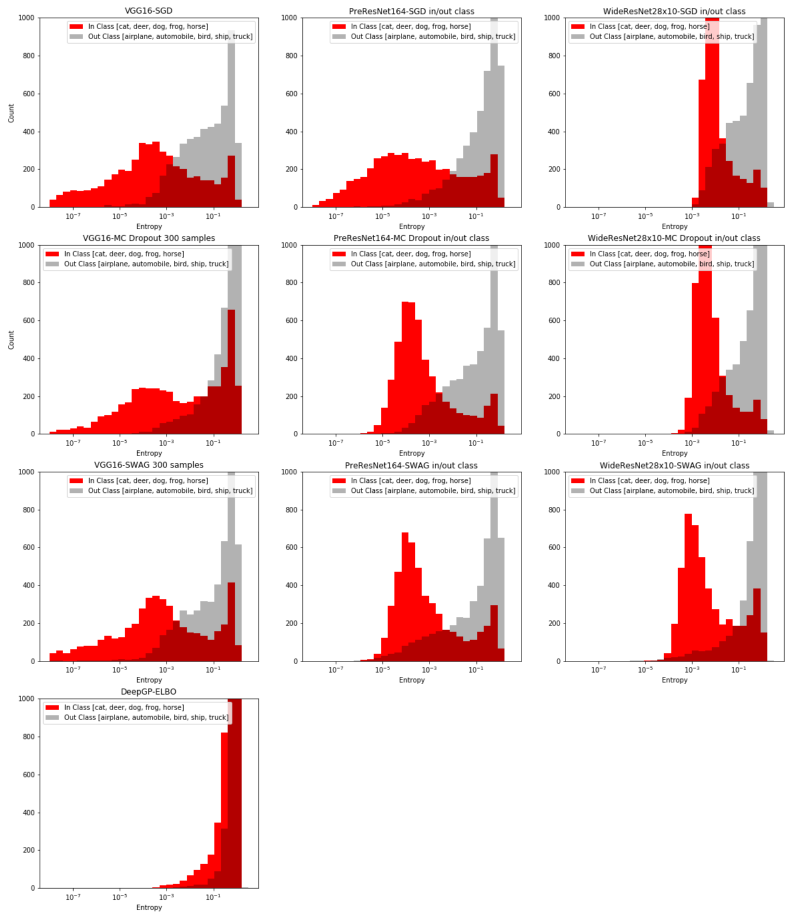

Fig. 2: Out of sample distributional entropy plots for all models on CIFAR-10 (5 + 5) categories.

7 Conclusion

In conclusion, as we have showed that point estimate deep neural networks indeed suffer from poor calibration and inability to identify out sample instances with high uncertainty. Bayesian deep neural networks provide a principled and viable alternative that allows the models to be informed about the uncertainty in their parameters and at the same time exhibits a lower degree of sensitivity against noisy samples compared to their point estimate DNN. This suggests that this is a promising research direction for improving the performance of deep neural networks.

Acknowledgement. This work was supported by Science Foundation Ireland under Grant No.15/CDA/3520 and Grant No. 12/RC/2289.

References

1. Choi, H., Jang, E., Alemi, A.A.: WAIC, but Why? Generative Ensembles for Robust Anomaly Detection. arXiv:1810.01392 [cs, stat] (Oct 2018)
2. Damianou, A.C., Lawrence, N.D.: Deep Gaussian Processes. arXiv e-prints (Nov 2012)
3. Fawzi, A., Fawzi, H., Fawzi, O.: Adversarial vulnerability for any classifier. Neural Information Processing Systems (Feb 2018)
4. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. arXiv e-prints (Jun 2015)
5. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On Calibration of Modern Neural Networks. International Conference on Machine Learning (Jun 2017)
6. He, K., Zhang, X., Ren, S., Sun, J.: Identity Mappings in Deep Residual Networks. arXiv e-prints (Mar 2016)
7. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging Weights Leads to Wider Optima and Better Generalization. arXiv:1803.05407 [cs, stat] (Mar 2018)
8. Krizhevsky, A.: Learning multiple layers of features from tiny images pp. 32–33 (2009), `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`
9. Kumar, A., Liang, P., Ma, T.: Verified Uncertainty Calibration. In: arXiv:1909.10155 [Cs, Stat]. vol. 33. Vancouver, Canada (Sep 2019)
10. Lee, K., Lee, H., Lee, K., Shin, J.: Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. arXiv:1711.09325 [cs, stat] (Nov 2017)
11. Maddox, W., Garipov, T., Izmailov, P., Vetrov, D., Wilson, A.G.: A Simple Baseline for Bayesian Uncertainty in Deep Learning. arXiv:1902.02476 [cs, stat] (Feb 2019)
12. Mandt, S., Hoffman, M.D., Blei, D.M.: Stochastic Gradient Descent as Approximate Bayesian Inference. arXiv e-prints (Apr 2017)
13. Nalisnick, E., Matsukawa, A., Teh, Y.W., Gorur, D., Lakshminarayanan, B.: Do Deep Generative Models Know What They Don't Know? International Conference on Learning Representations (2019)
14. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011), `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`
15. Polyak, B., Juditsky, A.: Acceleration of Stochastic Approximation by Averaging. SIAM Journal on Control and Optimization 30(4), 838–855 (jul 1992)
16. Ruppert, D.: Efficient Estimations from a Slowly Convergent Robbins-Monro Process. Technical Report TR000781, Cornell (feb 1988)
17. Schulam, P., Saria, S.: Can You Trust This Prediction? Auditing Pointwise Reliability After Learning. In: Proc. of Artificial Intelligence and Statistics. p. 10 (2019)

18. Shafaei, A., Schmidt, M., Little, J.J.: Does Your Model Know the Digit 6 Is Not a Cat? A Less Biased Evaluation of "Outlier" Detectors. arXiv:1809.04729 [cs, stat] (Sep 2018)
19. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv e-prints (Sep 2014)
20. Stracuzzi, D.J., Darling, M.C., Peterson, M.G., Chen, M.G.: Quantifying Uncertainty to Improve Decision Making in Machine Learning. Tech. Rep. SAND2018-11166, 1481629, Sandia National Laboratories (Oct 2018)
21. Wilson, A.G., Hu, Z., Salakhutdinov, R., Xing, E.P.: Stochastic Variational Deep Kernel Learning. arXiv e-prints (Nov 2016)
22. Zagoruyko, S., Komodakis, N.: Wide Residual Networks. arXiv e-prints (May 2016)