Known Unknowns: Uncertainty Quality in Bayesian Neural Networks

Ramon Oliveira* **Pedro Tabacof*** **Eduardo Valle**
RECOD Lab. — DCA / School of Electrical and Computer Engineering (FEEC)
University of Campinas (Unicamp)
Campinas, SP, Brazil
`{roliveir, tabacof, dovalle}@dca.fee.unicamp.br`

Abstract

We evaluate the uncertainty quality in neural networks using anomaly detection. We extract uncertainty measures (e.g. entropy) from the predictions of candidate models, use those measures as features for an anomaly detector, and gauge how well the detector differentiates known from unknown classes. We assign higher uncertainty quality to candidate models that lead to better detectors. We also propose a novel method for sampling a variational approximation of a Bayesian neural network, called One-Sample Bayesian Approximation (OSBA). We experiment on two datasets, MNIST and CIFAR10. We compare the following candidate neural network models: Maximum Likelihood, Bayesian Dropout, OSBA, and — for MNIST — the standard variational approximation. We show that Bayesian Dropout and OSBA provide better uncertainty information than Maximum Likelihood, and are essentially equivalent to the standard variational approximation, but much faster.

1 Introduction

While current Deep Learning focuses on point estimates, many real-world applications require a full range of uncertainty. Reliable confidence on the prediction might be as useful as the prediction itself. The debate over the dangers of overconfident machine learning has reached the headlines of mass media [1, 2]. Indeed, if our models are to drive cars, diagnose medical conditions, and even analyze the risk of criminal recidivism, unreliable confidence appraisal may have dire consequences.

Traditional Deep Learning trains by maximum likelihood — needing aggressive regularization to avoid overfitting — and only provides point estimates, with limited uncertainty information. If the model outputs a vector of probabilities (as a softmax classifier does), we can quantify its uncertainty using the entropy of the prediction. However, the model can predict with high confidence for samples way outside the distribution seen during training [3]. Frequentist mitigations, like the bootstrap [4], do not scale well for deep models.

True Bayesian models infer the posterior distribution over all unknown factors, but their computational demands are often prohibitive. On the other hand, we may profitably reinterpret under a Bayesian perspective some of the ad hoc regularizations used in ordinary Deep Learning (e.g., dropout [5, 6], early stopping [7], or weight decay [8, 9]). Gal and Ghahramani [5] show that multiple dropout forward passes in test time are equivalent to a Bayesian prediction (marginalized over the parameters' posteriors) given a particular variational approximation. A more direct (and expensive) approach variationally approximates the posterior of each weight [9].

*The first two authors contributed equally to this work.

2 One-Sample Bayesian Approximation (OSBA)

Here we propose a novel Bayesian approach for neural networks, similar to the variational approximation of Blundell *et al.* [9], but much cheaper computationally. We call that approach *One-Sample Bayesian Approximation* (OSBA), and investigate whether it achieves better quality of uncertainty information than traditional maximum likelihood.

We use exactly the same approach presented by Blundell *et al.* ([9], section 3.2), but instead of sampling the weight matrices for each training example, we sample the matrices only once per mini-batch, and use the same weights for all examples in that mini-batch. That approach leads to the same expected gradient, trading off higher variance for computational efficiency (about 10 times faster with a mini-batch of 100).

3 Uncertainty Quality

To evaluate the quality of uncertainty information, we employ anomaly detection: deciding whether or not a test sample belongs to the classes seen during training. More concretely, we pick a classification problem, exclude some classes from training, and use them to evaluate how much insight a candidate model has about its own classification confidence. We expect Bayesian neural networks to express such uncertainty well, to the point we can use it to decide whether a sample belongs or not to the known classes. Thus, we employ the AUC of the anomaly detector as a *relative* measure of the quality of the uncertainty information output by candidate models (Figure 1).

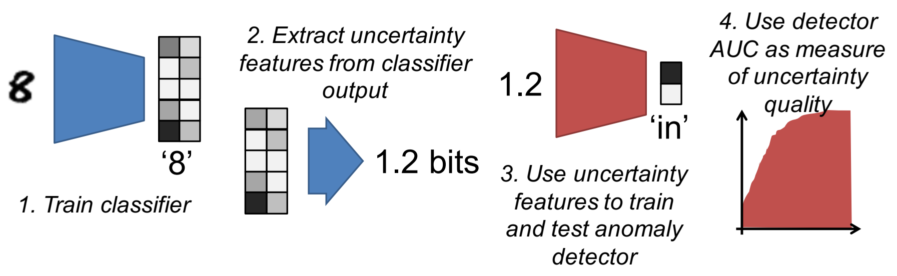

8
'8'
1. Train classifier
2. Extract uncertainty features from classifier output
1.2 bits
1.2
'in'
3. Use uncertainty features to train and test anomaly detector
4. Use detector AUC as measure of uncertainty quality

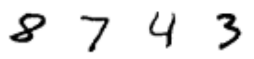
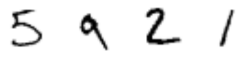
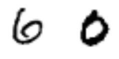
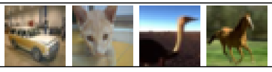
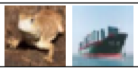
Figure 1: Uncertainty quality evaluation using an anomaly detection task. This is the experimental pipeline we follow to compare uncertainty quality among candidate models. (1) We train a candidate probabilistic classifier for the original task (MNIST or CIFAR10). (2) We extract uncertainty information from the classifier prediction. (3) We train a linear anomaly detector using those uncertainty measures as features. (4) We calculate the AUC of the anomaly detector. Higher detector AUCs indicate that a candidate model provides better uncertainty information.

We contrast two experimental protocols. In the Blind Protocol, we separate the classes into two groups (In and Out); train the candidate neural network using only the In classes; and then train — over the In *vs.* Out classes — a separate anomaly detector using the uncertainty extracted from the prediction of the candidate network. In the Calibrated Protocol, we separate the classes into three groups (In, Unknown, and Out); train the candidate network using the In classes with the loss function using the correct labels, and the Unknown classes with the loss function using the equiprobable prediction vector; and then train — over the In *vs.* Out classes — a separate anomaly detector using the same features as before. The test set used to compute the AUC of the anomaly task excludes (obviously) all samples used to train the anomaly detector, and (perhaps less obviously) all samples used to train the candidate neural network.

4 Methodology

We use MNIST [10] and CIFAR10 [11] datasets. For MNIST, the candidate networks have a two-layered fully-connected architecture with 512 neurons each, with dropout of 0.5 applied after each hidden layer. For CIFAR10, the candidate networks have two convolutional blocks (with dropout of 0.25 after each of them), followed by a fully-connected layer with 512 neurons (with dropout of 0.5). We optimize with ADAM [12], and limit each training procedure to 100 epochs for MNIST, and 200 epochs for CIFAR10. For each dataset we choose 4 In classes, 4 Out classes, and (for the Calibrated Protocol) 2 Unknown classes (Table 1). We randomize 20 combinations of In$\times$Out[$\times$Unknown] classes, with 5 repetitions each, totaling 100 replications.

Table 1: Possible combination for In, Out and Unknown classes, showing one sample per class. MNIST's classes have crisp semantic separation; CIFAR10's have considerable overlap due to specialization (e.g., animals) or to background (e.g., sky, lawn, pavement). Such overlap might reduce the accuracy of anomaly detection as a measure of uncertainty quality.

Dataset	In	Out	Unknown
MNIST			
CIFAR			

As methods, we evaluate the usual baseline of Maximum Likelihood (ML), a Bayesian posterior estimated from dropout [5, 13] (BD), our approximation for the standard variational Bayesian neural networks using one sample per mini-batch (OSBA), and, for MNIST, we also evaluate the standard variational approximation [9] (SV).

The features for anomaly detection are uncertainty measures extracted from probabilistic predictions. For simplicity, the detector is a linear logistic classifier, with regularization parameter set by stratified cross-validation [14]. For ML, only the vector of predicted probabilities is available, and thus we employ as feature the entropy — the most theoretically sound measure of uncertainty — over that vector. All Bayesian methods provide extra information; we use as feature vector the average and standard deviation of the entropy of the decision vector over 100 network prediction samples (estimating the expectation and variance of the entropy), the entropy of the average decision vector over those same samples (entropy of estimated expected predictions), and the average (over classes) of the standard deviations (over samples) of the predictions for each class.

4.1 Bayesian ANOVA

We analyze the results using Bayesian ANOVA [15], with a separate mean for each protocol (Blind *vs.* Calibrated). That is equivalent to a two-way ANOVA without interactions, where the global mean and experimental protocol factors are fused together (for interpretability). The methods (ML, BD, OSBA) are the factors of variation. We constrain the sum of the effects to be zero, for identifiability. The response variable is the logit-AUC of the anomaly detector. For easier interpretation, the results are transformed back into the AUC domain via the expit function. We use weakly informative priors. The following model reflects those choices:

$$\begin{aligned}
model &= \{ML, BD, OSBA\} \\
protocol &= \{Blind, Calibrated\} \\
logit(AUC_{protocol,model}) &\sim \mathcal{N}(\mu_{protocol} + \theta_{model}, \sigma) \\
\mu_{protocol} &\sim \mathcal{N}(0, 10) \\
\theta_{model} &\sim \mathcal{N}(0, \sigma_{theta}) \\
\sigma_{theta} &\sim \text{Half-Cauchy}(0, 10) \\
\sigma &\sim \text{Half-Cauchy}(0, 10) \\
\sum_{i \in model} \theta_i &= 0
\end{aligned}$$

We implement the model using Stan [16], and infer the posteriors of the unknown parameters using the NUTS algorithm [17]. To ensure proper convergence, we use 4 chains with 100K steps, including a 10K burn-in, and a thinning factor of 5. From Kruschke's suggestion [15], we present both the distribution of the marginal effects, and the distribution of the differences between effects.[1]

[1]Code for models, experiments, and analyses at `https://github.com/ramon-oliveira/deepstats`.

5 Results

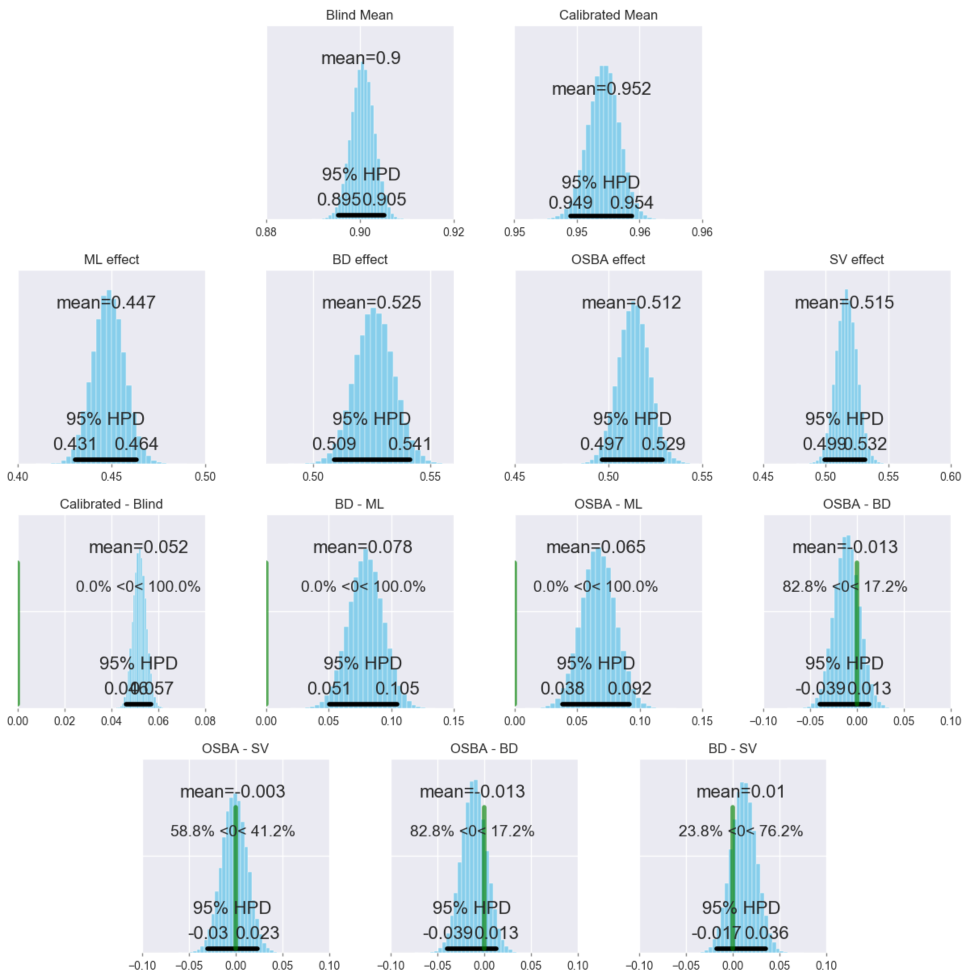


Figure 2: MNIST dataset. Each cell plots the distribution of the influence of the factor shown in the label above it, marginalized over all other factors. We highlight means (expected influence), and 95% Highest Posterior Density intervals (HPD, black bars). On the topmost two rows, we consider the factors themselves (marginal effect), and on the other rows, the differences between effects. We consider the differences significant if the HPD does not contain 0.0 (green bar). The domain is the AUC of the anomaly detector.

We show the Bayesian ANOVA results in Figures 2 and 3 (for reference, we also show the raw distributions of the AUCs, as boxplots in Figures 4 and 5). Calibration with the auxiliary Unknown classes has a large effect, larger than choosing among uncertainty methods. Calibration, however, is not realistic for many applications, due to the artificial constraint of picking well-formatted Unknown classes. On the well-controlled scenario provided by MNIST, Bayesian methods give significantly better uncertainty information than ML. On MNIST, all Bayesian methods outperform ML, and their effects do not appear significantly different from each other. On CIFAR10, however, perfect semantic separation between classes is questionable (Table 1), and the performance differences disappear: BD slightly outperforms ML, and OSBA slightly outperforms ML, but none of the differences appear significant.

Table 2 shows the accuracies of all candidate models. Note that competing candidate models have very similar performance: any gains in anomaly detection rather come from enhanced probabilistic information than from increased accuracy.

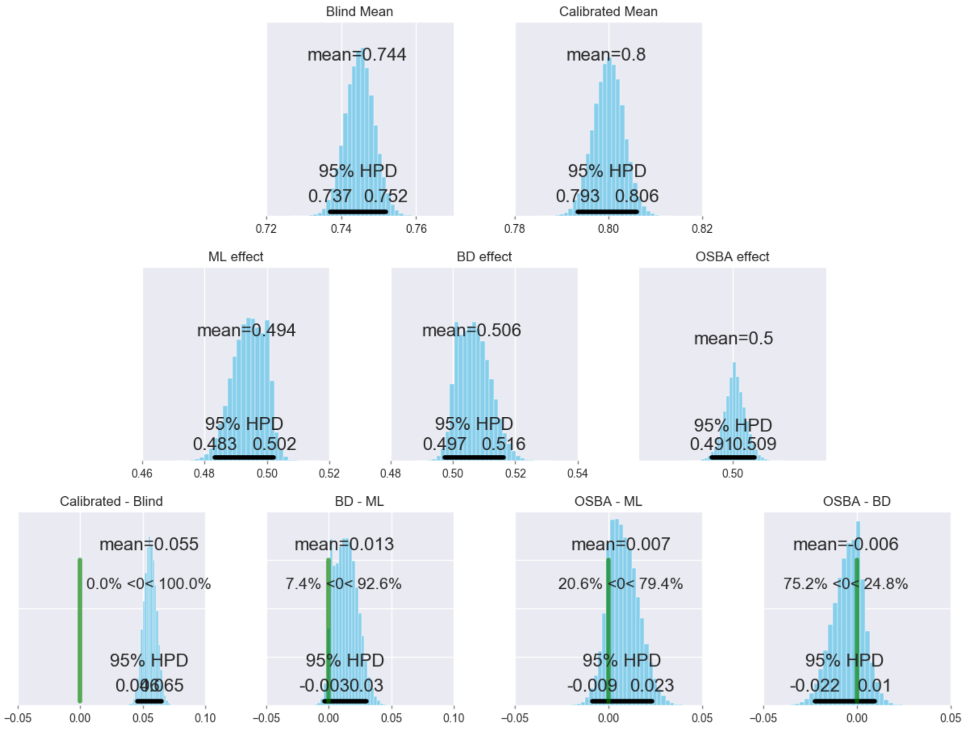


Figure 3: CIFAR10 dataset. Same information and interpretation as Figure 2 above.

Table 2: Test accuracy on the original classification task. We show the mean accuracy, with the standard deviation in parentheses, averaged over 100 different replications. Competing candidate models have similar accuracies, showing that enhanced uncertainty quality comes from enhanced probabilistic information, not from extra accuracy. Note that OSBA and SV have the same accuracy, but the latter is ten times slower.

Dataset	Protocol	ML	BD	OSBA	SV
MNIST	Calibrated	0.990 (0.002)	0.991 (0.002)	0.991 (0.002)	0.991 (0.002)
MNIST	Blind	0.992 (0.002)	0.992 (0.002)	0.991 (0.002)	0.991 (0.002)
CIFAR10	Calibrated	0.878 (0.036)	0.896 (0.033)	0.884 (0.037)	—
CIFAR10	Blind	0.905 (0.029)	0.908 (0.028)	0.896 (0.032)	—

6 Conclusion

We formalized how to ascertain uncertainty quality of neural networks by using anomaly detection. We contrasted the usual maximum likelihood networks to Bayesian alternatives. Bayesian networks outperformed the frequentist network in all cases.

We also proposed a novel way to sample from a variational approximation of a Bayesian neural network, OSBA, which is much faster than the standard sampling procedure, but still retains the same uncertainty quality. OSBA is 10× faster than SV; in our experiments, we observed relative training computational costs of 1× (ML) to 1× (BD) to 3× (OSBA) to 30× (SV).

We believe, thus, that techniques like BD and OSBA deserve further investigation in more contexts. Finding a general measure of uncertainty quality is, however, still a challenge. Our experiments suggest that anomaly detection only gives good uncertainty measures for well-separated classes, like MNIST's; for uncontrolled datasets like CIFAR10 (or ImageNet), we need a measure that tolerates a degree of semantic intersection between the classes.

As future work, we intend to explore other forms of uncertainty quality evaluation, and to test OSBA in more varied settings.

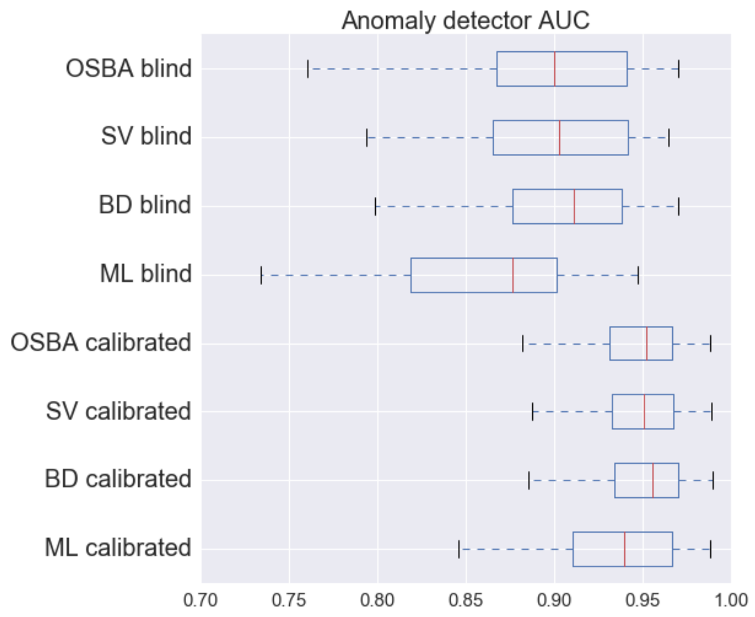


Figure 4: Distributions of the AUCs on MNIST for all combinations of probabilistic approach × experimental protocol. Each boxplot represents 100 replications, obtained by picking at random the In, Out, and (for the calibrated protocol) Unknown classes.

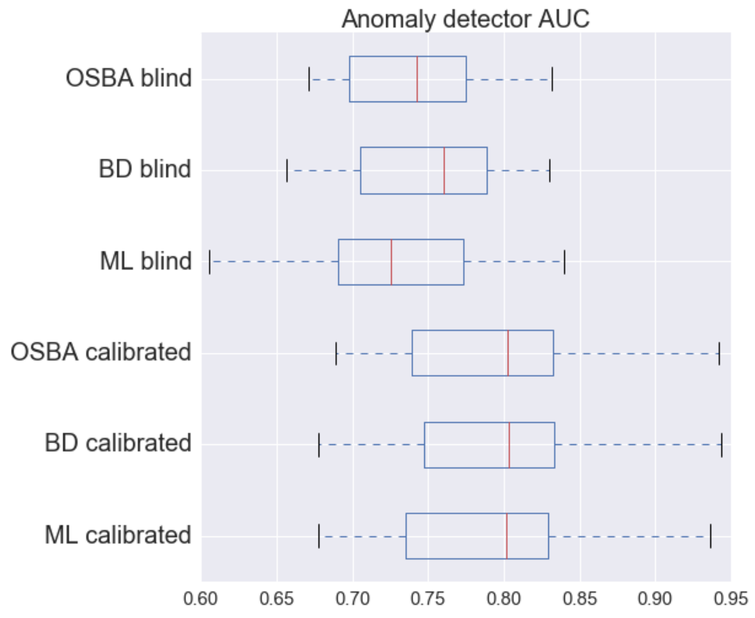


Figure 5: Distribution of the AUCs on CIFAR10, obtained the same way as Figure 4 above.

Acknowledgments

We thank Brazilian agencies CAPES, CNPq and FAPESP for financial support. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. Eduardo Valle is partially supported by a Google Awards LatAm 2016 grant, and by a CNPq PQ-2 grant (311486/2014-2). Ramon Oliveira is supported by a grant from Motorola Mobility Brazil.

References

[1] Jennifer Vaughan and Hanna Wallach. The inescapability of uncertainty: Ai, uncertainty, and why you should vote no matter what predictions say. `https://medium.com/@jennwv/uncertainty-edd5caf8981b`, 2016.

[2] Kate Crawford. Artificial intelligence's white guy problem. *The New York Times*, 2016.

[3] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[4] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.

[6] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.

[7] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Early stopping is nonparametric variational inference. *arXiv preprint arXiv:1504.01344*, 2015.

[8] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[9] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1613–1622, 2015.

[10] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[11] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

[14] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In *Encyclopedia of database systems*, pages 532–538. Springer, 2009.

[15] John Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014.

[16] Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Michael A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *J Stat Softw*, 2016.

[17] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.