

EFFICIENT VARIATIONAL BAYESIAN NEURAL NETWORK ENSEMBLES FOR OUTLIER DETECTION

Nick Pawlowski*
Imperial College London
np716@ic.ac.uk

Miguel Jaques*
No affiliation
migjaques@gmail.com

Ben Glocker
Imperial College London
b.glocker@ic.ac.uk

ABSTRACT

In this work we perform outlier detection using ensembles of neural networks obtained by variational approximation of the posterior in a Bayesian neural network setting. The variational parameters are obtained by sampling from the true posterior by gradient descent. We show our outlier detection results are better than those obtained using other efficient ensembling methods.

1 INTRODUCTION AND RELATED WORK

In this work we use posterior sampling by gradient descent in order to estimate the parameters of a variational approximation of the true posterior. This allows us to create an ensemble of networks that we use to perform outlier detection. We show that our method achieves better results than other efficient ensembling methods, though worse than standard ensembles. Our work is inspired by a mix of scalable ways to perform Bayesian deep learning Graves (2011); Blundell et al. (2015); Hernández-Lobato & Adams (2015); Gal & Ghahramani (2015); McClure & Kriegeskorte (2016), the use of ensembles to improve accuracy and obtain uncertainty estimates of the network’s predictions Zhou et al. (2002); Lakshminarayanan et al. (2016), and the ability to sample from the true posterior using stochastic gradient descent-like techniques Welling & Teh (2011); Teh et al. (2016). We use these samples to obtain a variational approximation of the true posterior, which can then be used to generate an ensemble of networks. We take inspiration from Huang et al. (2016), where weight samples are taken along the optimization trajectory, and used as an ensemble at test time. Instead, we take weight samples and use them to incrementally estimate the parameters of a variational approximation to the posterior during training, which allows us to sample from a distribution over weights while keeping constant memory requirements.

In Lakshminarayanan et al. (2016), the authors train N independent networks from scratch (with respective uncertainty estimates for regression) and use the ensemble to obtain good uncertainty estimates and classification performance. We view this model as an upper bound on how good our model can be, since the goal is to achieve similarly good results training only a single network.

This work is most comparable to *Dropout-MC* Gal & Ghahramani (2015) and other Dropout-related methods McClure & Kriegeskorte (2016). This is because only a single network is trained and the ensembles are only created for testing.

2 VARIATIONAL BAYES

In Bayesian inference we are interested in evaluating the predictive distribution

$$p(y|x, D) = \int p(y|x, w)p(w|D)dw \quad (1)$$

where $p(w|D) = p(D|w)p(w)/p(D)$ is the posterior over the weights, $D = (x^i, y^i)_{i=1}^M$ is the training data, $p(D|w)$ is the model likelihood, parametrized by a neural network with e.g. a softmax or Gaussian output, and $p(w)$ is the prior. In order to evaluate the integral above we first find an approximate posterior $q_\theta(w)$, where θ are the parameters of q (for example, if q is Gaussian, $\theta = (\mu, \sigma)$) and then use Monte-Carlo samples from this posterior to obtain an ensemble that can be used to estimate (1). In variational approximations the optimal parameters θ^* are usually

*Equal contribution.

found by minimizing the KL-divergence between the approximate and the true posterior. However, this form underestimates the variance of the posterior, hence its uncertainty. Since we are interested in a robust estimate of the uncertainty, we instead minimize the forward KL-divergence:

$$\theta^* = \operatorname{argmin}_{\theta} KL(p(w|D)||q_{\theta}(w)) = \operatorname{argmin}_{\theta} \mathbb{E}_{p(w|D)} [-\log q_{\theta}(w)] \quad (2)$$

Using a diagonal Gaussian as approximate posterior, $\log q_{\theta}(w) = \sum_i \log q_{\theta_i}(w_i) = \sum_i -\log \sigma_i - \frac{1}{2\sigma_i^2}(w_i - \mu_i)^2 + D$, where D is a constant, the minimization with respect to μ and σ^2 yields

$$\mu^* = \mathbb{E}_{p(w|D)}[w] \quad , \quad \sigma^{*2} = \mathbb{E}_{p(w|D)}[(w - \mu)^2] \quad (3)$$

This shows the optimal approximate weight means and variances are simply the first and second moments of the weights over the posterior distribution. In order to sample from the intractable posterior distribution of the weights without additional computation or memory requirements besides the standard stochastic gradient descent calculations, we can use Stochastic Gradient Langevin Dynamics (SGLD) Welling & Teh (2011); Teh et al. (2016). We use the trajectory of the weights through parameter space during training as samples from the unnormalized posterior $p(D|w)p(w)$, and compute μ and σ^2 using the standard unbiased Monte-Carlo estimates $\hat{\mu} = \sum_s^S w^{(s)}/S$ and $\hat{\sigma}^2 = \sum_s^S (w^{(s)} - \hat{\mu})^2 / (S-1)$, for $w^{(s)} \sim p(D|w)p(w)/Z$. Due to the large number of parameters in a neural network, storing all the samples (one per mini-batch step) would be too memory expensive. Instead we use the incremental formulas for computing $\hat{\mu}$ and $\hat{\sigma}^2$ from Welford (1962). This incremental estimation reduces the memory requirements to a minimum since we only have to store a single set of parameters during training. We can use this method as long as our training trajectory works as a valid sampling procedure, i.e. $w^{(s)} \sim p(D|w)p(w)/Z$, which is guaranteed by SGLD.

3 ENSEMBLES AND OUTLIER DETECTION

At test time, we sample weights from $q_{\theta}(w)$ in order to obtain a neural network ensemble and calculate the predictive probability using a Monte-Carlo estimate of (1). The ensemble is used to calculate an uncertainty score of the prediction. We use *disagreement* as defined by Lakshminarayanan et al. (2016) as uncertainty measure. The disagreement is defined as the sum of the KL-divergence between the average ensemble prediction and each ensemble component. We use the calculated disagreement to perform outlier detection. Therefore, we calculate the mean $\mu(d_{train})$ and standard deviation $\sigma(d_{train})$ of the disagreement d_{train} for a random training batch. A given sample is labelled as an outlier by thresholding:

$$p(\text{outlier} | x) = \begin{cases} 1, & \text{if } \mu(d_{train}) + 3\sigma(d_{train}) \geq d_x \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Here, the addition of $3\sigma(d_{train})$ to the threshold is used to minimize the false positive rate.

4 EXPERIMENTS AND RESULTS

In the experiments we used a 3-layer fully-connected neural network with 200 hidden units per layer and 10-unit softmax output, and an uninformative prior. We trained on the MNIST dataset and run outlier detection on the MNIST LeCun et al. (1998) and notMNIST Bulatov (2011) test sets. The networks were trained with a batch size of 100 for 500 steps. We compare our method against Snapshot Ensembles Huang et al. (2016), Dropout-MC Gal & Ghahramani (2015) and standard ensembles Lakshminarayanan et al. (2016) where each network is trained from scratch.

Although SGLD imposes some conditions on the step sizes and gradient noise (such as annealing) in order to guarantee that our procedure is a valid sampler from the true posterior, we found these hard to tune. Instead, using an Adam optimizer Kingma & Ba (2014) with fixed learning rate (0.005) and added gradient noise with fixed standard deviation (0.01) we obtained good results, though losing theoretical guarantees.

Table 1 shows the classification accuracies for the different models given the number of ensemble components. We were not able to reproduce the performance gain from snapshot ensembles as

Table 1: MNIST classification accuracy of different methods with various ensemble sizes. Standard ensembles perform best. Our method performs comparable to Dropout-ensembles but does not improve with higher ensemble size. The performance of snapshot ensembles was not reproduced.

Ensemble size	Standard	Snapshot	Dropout	Ours
1	4.00 ± 0.08	5.78 ± 0.09	5.24 ± 0.07	5.16 ± 0.18
5	2.92 ± 0.05	6.97 ± 0.16	4.35 ± 0.23	4.97 ± 0.18
10	2.67 ± 0.06	7.12 ± 0.15	4.29 ± 0.16	5.06 ± 0.15

presented by Huang et al. (2016). This could be caused by the use of an easier task or the small number of steps. The classification performance of our method is comparable to the performance of Dropout-MC and a simple neural network. Standard ensembles perform best. However, the performance of our method does not increase with the number of components in the ensemble. Analysing the disagreement results across the ensembles shows very small disagreement. Our method shows disagreements of order 10^{-3} to 10^{-2} , whereas standard ensembles have a disagreement of 10^{-1} to 10^0 . This means that the variational approximation fits a Gaussian with small variance. However, outlier detection depends on relative rather than absolute disagreement values between in-dataset samples and out-of-dataset samples.

We perform outlier detection on the test set of both MNIST and notMNIST. The results for various ensemble sizes can be seen in Fig. 1. Our method shows very low false positive rates which are comparable to Dropout-MC and Snapshot Ensembles, but outperforms both of them for true positives. Only real ensembles achieve higher true positive rates for outlier detection. This can be explained by the use of different initialisations which lead to a higher variety in trained models.

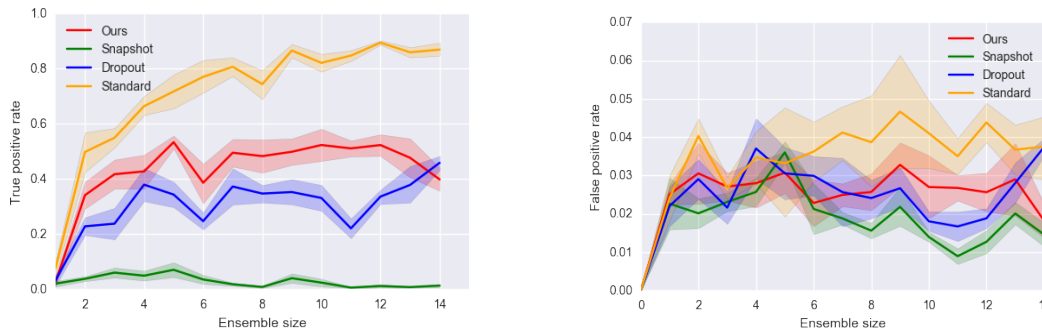


Figure 1: True and false positive rates for outlier detection using different ensembling methods and various ensemble sizes. The methods are tested on the test sets of MNIST and notMNIST (outlier). All methods show low false positive rates. Our method achieves a good true positive rate and is only outperformed by standard ensembles.

5 CONCLUSION

We present an efficient method for building variational approximations of neural networks. It enables the creation of ensembles. Our method is easily adaptable to a wide range of neural network architectures, provides comparable prediction performance and only minimally increases computation and memory requirements during training. Additionally, we propose a simple ensemble based outlier detection method which is inspired by Lakshminarayanan et al. (2016).

ACKNOWLEDGMENTS

NP is supported by Microsoft Research through its PhD Scholarship Programme and the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1).

REFERENCES

- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.
- Yaroslav Bulatov. NotMNIST dataset. *Google (Books/OCR), Tech. Rep.[Online]. Available: <http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html>*, 2011.
- Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. 2015.
- Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.
- José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, pp. 1861–1869, 2015.
- Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *OpenReview: ICLR 2017 conference submission*, 2016.
- Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.
- Yann LeCun, Corinna Cortes, and Christopher JC Burges. The MNIST database of handwritten digits, 1998.
- Patrick McClure and Nikolaus Kriegeskorte. Representation of uncertainty in deep neural networks through sampling. *arXiv preprint arXiv:1611.01639*, 2016.
- Yee Whye Teh, Alexandre H Thiery, and Sebastian J Vollmer. Consistency and fluctuations for stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 17(7):1–33, 2016.
- BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.
- Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.
- Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.