Josef Haddad

# Essay on Uncertainty Estimation

## Introduction

In this essay, I will review and compare three papers that present approaches for uncertainty estimation with deep networks. Uncertainty estimation with deep networks is worth studying because of the wide variety of tasks that deep networks have shown to be capable of helping with. Taking uncertainty into consideration is important to prevent having overconfident predictors. A model that doesn't express uncertainty in its predictions can be dangerous, and a model can be uncertain in scenarios that it hasn't been trained on (out-of-distribution examples). It could, for example, be a self-driving drone that is flying in never before seen conditions. With a model that can express uncertainty, one can avoid drone crashes by alerting the pilot on the ground to take over the steering when it is uncertain about how to behave, instead of just making the action that it is most certain about.

Uncertainty estimation in autonomous models will not only benefit drone flying but self-driving vehicles overall. It will also benefit the area of medicine with predictions about health conditions, which can be used to alert a professional when the model is uncertain about a decision. Being able to incorporate such models would save time in the medical field, and potentially save lives.

Previous work done in [MacKay, 1992] includes Bayesian approaches with neural networks where the maximum number of parameters used in their neural network models is 25. This is previous work that the three selected papers mention directly or indirectly. The papers I will summarize have their ways of incorporating uncertainty with deep networks, which are presented in the method section.

## Method

The first paper "Weight Uncertainty in Neural Networks" [Blundell et al., 2015], introduce uncertainty to the weights of deep networks as a way to combat overfitting and as a way of incorporating uncertainty in the data. They further present experiments for classification, regression and on-line learning in reinforcement problems. Each weight in their networks is assigned a Gaussian distribution with individual mean $\mu$ and variable $\rho$, which is a parameter for calculating the standard deviation $\sigma$ of the weights. To train the parameters $\theta$ ($\mu$ and $\rho$) that each weight $\mathbf{w}$ has, they propose "Bayes by Backprop". [Blundell et al., 2015] use variational learning of $\theta$. Their loss function is, therefore, the Kullback-Leibler divergence from $P(\mathbf{w}|D)$ to $q(w|\theta)$, where D is our data, and q denotes a probability density. They use an approximate form of this loss. They use Monte Carlo samplings of $\mathbf{w}$ (drawn from Gaussian($\mu$, $\rho$)) to get an estimate of the gradients to minimize the loss function in an SGD fashion on every $\mu$ and $\rho$, which is necessary due to how computationally inefficient the model training would be if the exact gradient was used. So to update the gradients, they sample weights, calculate the gradients with respect to every $\mu$ and $\rho$ in the network (for each weight), and then repeat by sampling new weights $\mathbf{w}$. For prediction, they sample an ensemble of networks which together express the uncertainty of the prediction.

The second paper, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles" [Lakshminarayanan et al., 2017], also handle predictive uncertainty for classifica-

tion and regression problems, but furthermore address the problem of difficulties in training Bayesian models, both from a practical and computational perspective. Lakshminarayanan et al. Train an ensemble of non-Bayesian networks that has output nodes corresponding to mean $\mu$ and variance $\sigma^2$ respectively. The models in the ensemble (typically around the default quantity of 5 models) are trained separately with the negative log-likelihood as the loss function. Each output of the model is weighted equally to form a Gaussian when using the ensemble for prediction. I will refer to their method as the "Deep Ensembles" approach in this essay.

The third paper addressed in this review is "A Simple Baseline for Bayesian Uncertainty in Deep Learning" [Maddox et al., 2019], where the authors, just like in the first paper [Blundell et al., 2015], approximate posterior distributions of the weights to handle uncertainty with deep networks. [Maddox et al., 2019] modify the training algorithm Stochastic Weight Averaging (SWA), and call their approach SWA-Gaussian (SWAG). Standard SWA in deep learning averages the final model weights over the weights that it has had during training. This average of each weight is used as the mean for the distribution of the weights in SWAG. SWAG additionally incorporates the seen weights to form a covariance matrix over the weights. This covariance matrix consists of a diagonal covariance matrix and a non-diagonal covariance matrix of the weights but with a lower rank which decreases computation and storage requirements. When applying their approach, one ideally starts from a pre-trained model and apply the SWAG approach to that. For prediction, Bayesian model averaging is performed by sampling multiple networks from our weight distribution. [Maddox et al., 2019] use 30 sampled networks in their experiments for prediction.

## Comparison

These three presented approaches yield different network characteristics. Bayes by Backprop and SWAG are similar in the way that they model distributions of the weight while the Deep Ensembles approach in [Lakshminarayanan et al., 2017] is non-Bayesian and is more similar to standard feedforward networks with constant weights but has output nodes which correspond to the mean and variance. The computational complexity of these models also vary but training time can, of course, be adjusted by stopping the training earlier. Bayes by Backprop requires doing Monte Carlo sampling of the whole network each time the gradient is calculated which is inefficient compared to the other methods. The efficiency of using the Deep Ensembles approach depends on how many networks one uses in the ensemble but with 5 networks (recommended default value) in the ensemble, it shouldn't be too bad. [Lakshminarayanan et al., 2017] furthermore use networks that are more similar to the more conventional standard feedforward networks making their method simpler to understand and implement.

Another aspect of complexity is the required memory. Introducing independent Gaussian distributions to every weight in the network doubles the number of parameters, and since we sample complete networks, we need additional memory for a whole new network making the requirement 3 times the number of weights if we sample the networks sequentially. This is the case for the Bayes by Backprop approach. SWAG at least doubles the number of weights but

also samples complete networks for prediction. SWAG further more stores a non-diagonal covariance matrix of rank K (K is a hyperparameter in the model and an integer larger than one), which requires memory of at least K times the number of weights in the model. [Maddox et al., 2019] use rank K = 20 in their experiments with SWAG, which means that the required memory is at least 20 times the number of weights when using the default setting. The required memory for the Deep Ensembles approach depends on the number of networks in the ensemble but with the recommended number of networks (five), the memory required becomes five times the number of weights of one network in the ensemble, and we do not have to sample new networks for training or prediction.

The experiments in [Blundell et al., 2015] and [Lakshminarayanan et al., 2017] showed that the Deep Ensemble performs better than the Bayes by Backprop approach when an ensemble of five networks is used for each network on the MNIST-dataset. This holds for both the use of a scale mixture Gaussian prior and a standard Gaussian prior of the model trained using Bayes by Backprop. [Lakshminarayanan et al., 2017] achieved a classification error of $\sim 1.14\%$ while [Blundell et al., 2015] achieved an error of 1.32% when using a scale mixture Gaussian as prior. The difference isn't very large and the two papers used different network architectures which makes the comparison unfair. [Lakshminarayanan et al., 2017] used 2 hidden layers with 1200-nodes each for their best performing model while [Blundell et al., 2015] used 3 hidden layers with 200-nodes each. [Maddox et al., 2019] did not test the SWAG approach for classification on MNIST-data but SWAG performs on par with models from previous work on the CIFAR-10 dataset (using the same network architecture) with the addition of being one of the better-calibrated models, at least shown on the CIFAR-100 dataset.

Considering the complexity of the different approaches, Deep Ensembles is the one which the broader deep learning audience would be able to understand and make use of. The training for that approach is very similar to what many learn in introductory deep learning courses or books. The major differences are that a softplus function is used for the output of the variance and the loss function used is the negative log-likelihood which some may be a bit unfamiliar with. Introducing posterior distributions to the weights of the networks is new to many and the approaches for training these that I have covered are more complex and harder to work with overall.

These reviewed approaches for incorporating uncertainty requires more resources in terms of both memory and computing power compared to standard neural networks. This makes them unsuitable for mobile devices, such as remote sensors on drones, where the components often are limited in terms of power consumption and weight. However, these approaches may be suitable if we prune the networks first and if the networks already are trained, so that we only have to handle prediction on those devices. This is possible for the model trained using Bayes by Backprop. There was only a minor decrease in performance when pruning a 2.4m weight model by 98% (to 48k weights) when using the Signal-to-Noise ratio as a metric for pruning. The error rate only increased to 1.39% for a model with 1.24% before pruning. If the prediction is fast enough for a self manoeuvring drone, then this would be very promising for quick decision making. It would be interesting to also try pruning the networks used in the other papers to see if the predictive power holds.

# References

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

[Lakshminarayanan et al., 2017] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413.

[MacKay, 1992] MacKay, D. J. (1992). Bayesian interpolation. *Neural computation*, 4(3):415–447.

[Maddox et al., 2019] Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13153–13164.