Overpruning in Variational Bayesian Neural Networks

Brian L. Trippe
University of Cambridge and
Massachusetts Institute of Technology
btrippe@mit.edu

Richard E. Turner
University of Cambridge
ret26@cam.ac.uk

Abstract

The motivations for using variational inference (VI) in neural networks differ significantly from those in latent variable models. This has a counter-intuitive consequence; more expressive variational approximations often provide significantly worse predictions as compared to those with less expressive families. In this work we make two contributions. First, we identify a cause of this performance gap, variational over-pruning. Second, we introduce a theoretically grounded explanation for this phenomenon. Our perspective sheds light on several related published results and provides intuition into the design of effective variational approximations of neural networks.

1 Introduction

Though deep neural networks have been enormously successful across a variety of prediction tasks, they often fail to accurately capture uncertainty, a characteristic which has motivated a resurgence of interest in Bayesian methods for learning neural networks. Following initial work on using variational inference(VI) to fit neural networks [6, 1], a great deal of recent work has proposed new approaches for VI in these models [5, 2, 4, 8, 11]. However, VI remains difficult and the performance benefits as compared to other approaches for capturing uncertainty is unclear [9].

The motivations for using VI to capture parameter uncertainty differ significantly from those for using VI in latent variable models for which we are inherently concerned with a posterior over hidden variables. In Bayesian approximations of neural networks, the posterior over weights and biases generally is not the object of interest; instead we are concerned with the posterior over functions. As a result, failing to capture characteristics of the exact posterior such as multi-modality (which we know to be imparted by the many symmetries and degeneracies of NNs) is not necessarily problematic. Empirically, methods that perform VI over parameters that we know not to even resemble the exact posterior over parameters can perform acceptably in practice, providing reasonably well calibrated uncertainty predictions on small datasets [4, 2].

The organization of this short paper is as follows. In section 2 we document a surprising consequence of this mismatch; more expressive variational approximations often provide worse performance than less expressive ones. Next, in section 3 we identify a cause of this performance gap, the over-pruning of hidden units. Finally, in section 4 we provide a theoretical explanation for this phenomenon which clearly explains over-pruning as well as several other peculiar results in variational Bayesian neural network literature.

2 Variational Approximations of Neural Networks

We consider supervised learning problems in which we have a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ of observation/label pairs, $(x_i \in X, y_i \in Y)$, sampled i.i.d. from some joint distribution, $p(x, y)$, and are interested in estimating the conditional, $p(y_{\text{new}}|x_{\text{new}})$. We suppose the labels are sampled from a

Advances in Approximate Bayesian Inference — NIPS 2017 Workshop — Long Beach, CA, USA.

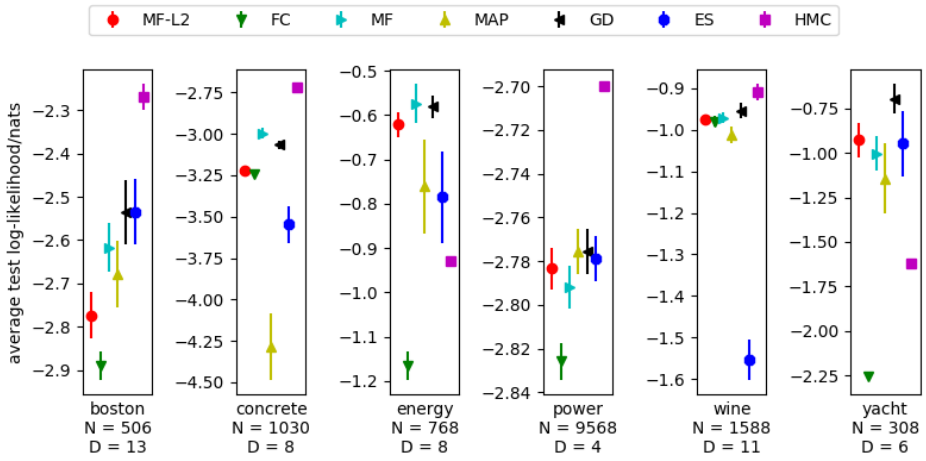


Figure 1: Comparison of performance Bayesian neural networks in terms of mean log likelihood on test sets using different variational families on six small UCI datasets. Higher is better. Uncertainty bars represent $\pm 1 SEM$.

discriminative probabilistic model parameterized by θ, such that $y_i \sim p(y|x_i, \theta)$ and use $\mathcal{D}$ to learn about θ in order make predictions.

The Bayesian approach considers θ to be an unknown variable, places a prior over it (here parameterised by α) and seek the posterior $p(\theta|\mathcal{D}, \alpha) = \frac{p(\theta|\alpha)p(Y|X,\theta)}{\int_{\hat{\theta}} p(\hat{\theta}|\alpha)p(Y|X,\hat{\theta})d\hat{\theta}}$. In our case, θ defines the parameters of a neural network, and we make predictions by approximating the marginal over θ with a Monte Carlo estimate:

$$p(y_{\text{new}}|x_{\text{new}}, \mathcal{D}, \alpha) = \mathbb{E}_{p(\theta|\mathcal{D},\alpha)}[p(y_{\text{new}}|x_{\text{new}}, \theta)] \approx \frac{1}{M}\sum_{i=1}^{M} p(y_{\text{new}}|x_{\text{new}}, \theta_i) \tag{1}$$

Where M is the number of Monte Carlo samples of $\theta_i \sim p(\theta|\mathcal{D}, \alpha)$. Variational inference (VI)[7] minimizes the KL-divergence between an approximation of the intractable posterior, $q(\theta)$, and $p(\theta|\mathcal{D}, \alpha)$:

$$\arg\min_{q\in\mathcal{Q}} \text{KL}[q||p] = \arg\min_{q\in\mathcal{Q}} \mathcal{F}_{\text{VFE}}(q) = -\mathbb{E}_{q(\theta)}\big[\log p(Y, \theta|X, \alpha) - \log q(\theta)\big] \tag{2}$$

The choice of variational family, $\mathcal{Q}$, is important when employing variational approximations of neural networks, and a number of different variational families have been proposed [5, 4, 11, 12]. We explore the performance of several variational families when applied to a single layer MLP with 50 hidden units and tanh activations on six benchmark UCI regression datasets. The methods include maximum likelihood inference with early stopping (ES), maximum a posteriori using a Gaussian prior over weights (MAP), variational inference with a mean-field Gaussian approximate posterior over weights with learned variances (MF)[8], mean-field Gaussian with fixed variances (GD)[1] and Gaussian approximate posterior with full rank covariance within each layer (FC). For each of these models we tuned hyper-parameters on a held-out validation set. Notably, this included the prior variance for MAP, GD, MF and FC, and additionally included weight and bias variances for GD. We additionally include the performance of two sampling methods, hybrid Monte Carlo (HMC)[14] as previously evaluated on these datasets by [3].

As we see in Figure 1, GD most consistently has good performance as compared to the other variational approximations tested. MF performs well on most datasets with best performance on concrete. ES and MAP have high variance their performance, both across tasks and across train/test splits within each task; each of these methods have the greatest performance on one dataset and the worst performance on at least one dataset.

[1] We refer to this model as Gaussian Dropout(GD) due to its equivalence to this model[8]. This model has been described as fast dropout [18].

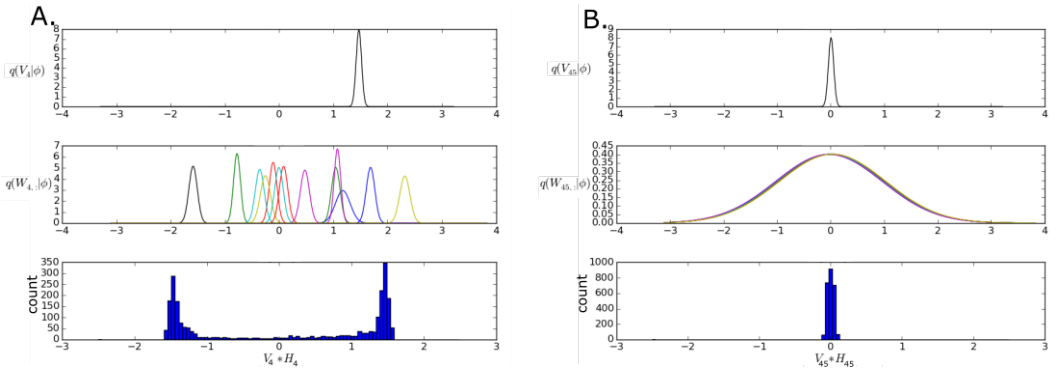


Figure 2: Representative examples of the characteristics hidden units in a neural network with tanh activation functions trained by variational inference with a factorized Gaussian approximate posterior. A.) an active hidden unit B.) a pruned hidden unit. Top.) the approximate posterior over the weight connecting it to the output. Middle.) posteriors over weights connecting from the input to the hidden unit Bottom.) Histogram of 25 sampled activations across all training data-points. These reflect learned weights for the same network discussed in Figure 4. Similar results for the remaining hidden units are included in the appendix (Figure 5).

FC performs worse than the mean-field approximation. This is the case on all datasets other than wine quality prediction, on which the two methods have roughly equivalent performance. This could be explained by the use of the local reparameterization trick [8] on MF and GD, which we were unable to use for FC, which used a more structured variational family.

Surprisingly, we see that richer variational approximations often perform worse than less flexible ones. We note that the family of approximations which can be represented by GD is a strict subset of those which can be represented by MF, which itself is a strict subset of those representable by FC. Though the more expressive families can achieve a lower variational free energy (or equivalently, a better expectation lower bound), this does not lead to better predictions[2].

3 Over-pruning in Mean Field Approximations

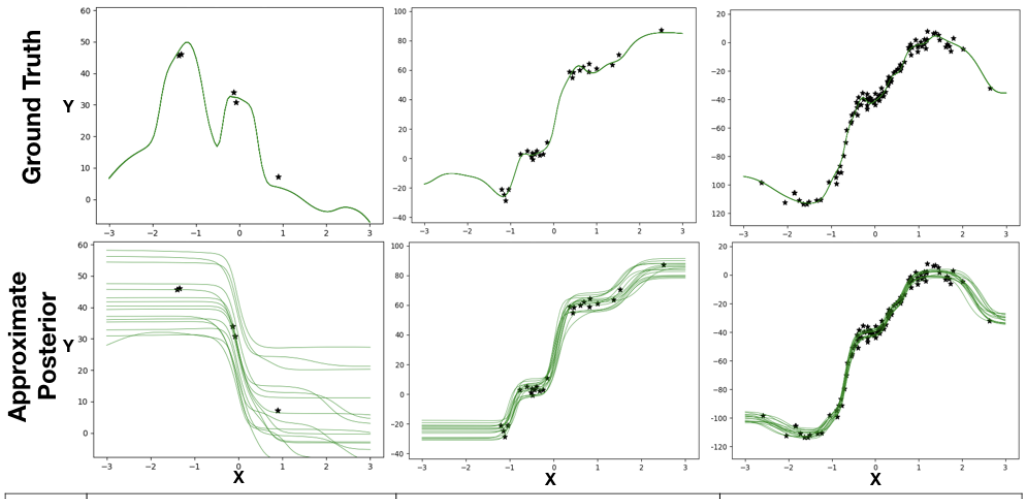
To better understand why the mean field variational family performs worse than the weight noise model, we took a closer look at a posterior approximation fit to the 'Boston Housing' dataset. Watching the trajectories of the KL divergence, expected negative log-likelihood and negative log-likelihood under the posterior predictive distribution, we make several observations (Appendix Figure 4). As anticipated, the estimated $\mathcal{F}_{\mathrm{VFE}}$ decreases monotonically throughout the course of optimization. The negative log-likelihood under the posterior predictive distribution, however, shows unexpected behavior. For both the training and test sets, this term initially decreases but then rebounds slightly, increasing and converging to larger a value than it had taken earlier. While an increase in the negative log expected likelihood of test data is a sign of overfitting, this behavior on the training set indicates underfitting.

Looking more closely at the posterior approximation we see that most of the hidden units have been pruned away (Figure 2). Of the 50 hidden units, 39 have learned output weight confidently around zero with all incoming weights sitting precisely at the prior, and the remaining 11 have learned outputs weights farther from zero with incoming weights that are more certain and dispersed. We refer to this phenomenon as variational overpruning and believe it is largely responsible for the performance decay seen in figure 2.

To investigate the possibility that the observed pruning was an artifact the optimization [3], more performed an experiment in a more simplified setting (Figure 3). We simulated data from a neural network by sampling weights and biases from a Gaussian prior with Gaussian observation noise, and

[2]The differently chosen priors for each of the models tested precludes an informative comparison of these bounds.

[3]This possibility initially seemed plausible given that the objective is non convex, and optimization relies of noisy gradient estimates



N	5	25	100
Units Pruned	47/50	43/50	37/50

Figure 3: Pruning in correctly specified models and induces a biased posterior over functions **Top.)** Data is simulated from from the a single layer 50 hidden unit MLP by sampling weights from the prior, mapping inputs through the function and adding observation noise. Three datasets are simulated, including 5, 25 and 100 points(from left to right) **Bottom.)** Pruning occurs robustly, and functions sampled from the learned approximate posteriors are biased, including steps corresponding to the inflection points of the 'tanh' activation function of each of the unpruned hidden units. This effect is less pronounced for larger N, but still leads to underfitting. The 'true' weights were used as the initial means in the variational approximation.

performed inference initializing the variational approximation with mean equal to the true parameters and a small initial variance. This construction allows us to observe the behavior of a correctly specified model.
As the uncertainty in the approximate posterior increases from its small initialization, the most hidden units are pruned away. As a result, functions within the support of the posterior consist of a small number of steps, each corresponding the tanh nonlinearity of an un-pruned hidden unit.

Overpruning limits the expressiveness of these models and leads to under-fitting, in both a toy example and in real regression problems. In the next section, we explain how this phenomenon is a manifestation of a more general problem with variational methods[16].

4 Tightness of the Variational Bound Explains Overpruning

In this section, we propose theoretically grounded explanation for overpruning. We can gain insight into the performance of the mean-field approximation by decomposing $\mathcal{F}_{\mathrm{VFE}}(q)$ into the expected log likelihood and a complexity penalty for each layer. This decomposition provides a clear explanation for the source of variational overpruning.

To be concrete, consider an MLP with a single hidden layer of N hidden units with activations denoted as $\mathbf{h} = (h_1, h_2, \ldots, h_N)$, defined as a function of weight matrix, W, and input, x, as $\mathbf{h} = \tanh(W \cdot x)$, and a single output defined as the dot product of a second weight matrix, V, with

$\mathbf{h}$ such that $f_\theta(x) = V \cdot \tanh(W \cdot x)$[4]. As such, we have parameters $\theta = (W, V)$, and can write the variational objective as:

$$\begin{aligned}\mathcal{F}_{\mathrm{VFE}} &= -\mathbb{E}_{q(\theta)}\big[\log p(Y|X,\theta)\big] + \mathrm{D}_{\mathrm{KL}}\big(q(\theta)||p(\theta|\alpha)\big) \\ &= -\mathbb{E}_{q(\theta)}\big[\log p(Y|X,\theta)\big] + \sum_{j=1}^{N} \mathrm{D}_{\mathrm{KL}}\big(q(v_j)||p(v_j|\alpha)\big) + \sum_{i=1}^{D} \mathrm{D}_{\mathrm{KL}}\big(q(w_{j,i})||p(w_{j,i}|\alpha)\big)\end{aligned} \tag{3}$$

This presentation of $\mathcal{F}_{\mathrm{VFE}}(q)$ as the sum of the expected log-likelihood and the complexity penalty(the KL-divergence of $q(\theta)$ from the prior, $p(\theta|\alpha)$) defines a trade-off between modeling the complexity of the data and retaining the simplicity of the prior [2]. The pruning of hidden units as in figure 2B reduces the tension of this trade-off. As we see in figure 2B, when the approximate posterior over a hidden-to-output weight, v_j, is centered on 0 with low uncertainty, the corresponding hidden unit, h_j, no longer impacts the output. In turn, the incoming weights to h_j, $w_{j,:}$ no longer have an impact on the expected log-likelihood. As a result:

$$p(w_{j,i}|v_j = 0, \mathcal{D}, \alpha) = \frac{p(w_{j,i}|v_j = 0, \alpha)p(\mathcal{D}|v_j = 0, \alpha)}{p(\mathcal{D}|v_j = 0, \alpha)} = p(w_{j,i}|\alpha)$$

In this way, learning variational approximations $q(v_j) \approx \delta(0)$, establishes conditional independence between each $w_{j,i}$ and the data, and $p(w_{j,i}|v_j = 0, \mathcal{D}, \alpha)$ collapses to its prior. In a network with a single output and multiple inputs, incoming weights far outnumber output weights and pruning provides a mechanism for reducing the complexity penalty without incurring a large penalty for increasing the variance in predictions (as would occur if the output weights were uncertain as well). This mechanism reduces the variational free energy by bringing the exact posterior closer to the prior rather than by explaining the data. This is reminiscent of the known property of variational maximizations that the tightness of the variational bound induces biases in parameter estimates [16].

We argue that variational over-pruning is a common pathology of variational approximations to neural networks in which variances are learned, and believe the effect of pruning is compatible with several surprising documented observations. For example, Blundell et al.[2] showed that up to 98% of the weights of a network trained on MNIST could be pruned with an accuracy decay of only 0.15% (from 1.24% error to 1.39%) [5]. A separate result which we believe is explained by variational over-pruning was published by Molchanov et al.[13], who showed that, when optimizing the dropout probabilities of networks using variational dropout, many drop probabilities drifted to 1. They referred to this property as inducing sparsity, which we believe to be an manifestation of pruning. More recently, [10] found this to occur in mean field Gaussian variational approximations, and used this property to significantly compress Bayesian networks. In our eyes, this is a shortcoming rather than a feature.

In contrast to MF, GD does not have learned variances and is therefore unable to prune hidden units. This encourages learning of parameters which define smoother functions and does not underfit, thus explaining the observed performance gap.

5 Conclusion

We have demonstrated a surprising property of variational approximations to neural networks; expressive approximations often provide worse performance than more constrained approximations. We identified variational overpruning as an explanation for this phenomenon and provided a theoretical explanation for why it occurs. Despite much recent work on improving variational approximations to neural networks, theoretical justification for the use of one family over another is largely absent and it is often unclear how to choose variational families. We hope our perspective provides a grounding for the selection of variational approximations.

6 Acknowledgments

The authors would like to thank Thang Bui, Yingzhen Li, and Cuong Nyguyen for insightful comments and discussion and the reviewers for constructive feedback.

[4] We neglect biases for simplicity. With biases, the argument is identical but notationally more complex.

[5] It is worth noting, however, that the performance of their approach did not surpass that of drop-connect which can be viewed as using a less expressive variational family[17]

References

[1] D Barber and C M Bishop. Ensemble learning for multi-layer networks. *Advances in Neural Information Processing Systems*, pages 395–401, 1998.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. *Icml*, 37:1613–1622, 2015.

[3] Thang D Bui, Daniel Hernández-Lobato, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. Deep Gaussian Processes for Regression using Approximate Expectation Propagation. *ICML*, 48, 2016.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning. *Icml*, 48:1–10, 2015.

[5] Alex Graves. Practical Variational Inference for Neural Networks. *Nips*, pages 1–9, 2011.

[6] Geoffrey E. Hinton and Drew Van Camp. Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. *ACM COLT*, 1993.

[7] Michael I Jordan, Tommi S Jaakkola, Lawrence K Saul, and Florham Park. An Introduction to Variational Methods for Graphical Models An Introduction to Variational Methods for Graphical Models. 233(January):183–233, 1998.

[8] Diederik P Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. *arXiv*, (Mcmc):1–13, 2015.

[9] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *arXiv*, 2017.

[10] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian Compression for Deep Learning, 2017.

[11] Christos Louizos and Max Welling. Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors. *Icml*, 48, 2016.

[12] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. 2017.

[13] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational Dropout Sparsifies Deep Neural Networks. *ICML*, 2017.

[14] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, 1995.

[15] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Adv. Neural Inf. Process. Syst. 25*, pages 1–9, 2012.

[16] Richard E Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time-series models. *Inference and Estimation in Probabilistic TimeSeries Models*, pages 109–130, 2011.

[17] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. *Icml*, (1):109–111, 2013.

[18] Sida I Wang and Christopher D Manning. Fast dropout training. *ICML*, 28:118–126, 2013.

Table 1: Held-out mean log-likelihood in nats $\pm 1SEM$ for several variational methods for fitting a 50 hidden unit neural network benchmark regression datasets. Higher is better.

Dataset	N	D	**HMC**	**SGLD**	**ES**	**FC**	**MAP**	**MF**	**GD**
boston	506	13	**-2.27±0.03**	-2.40±0.05	-2.53±0.08	-2.89±0.03	-2.68±0.08	-2.62±0.06	-2.54±0.07
concrete	1030	8	**-2.72±0.02**	-3.08±0.03	-3.55±0.11	-3.24±0.01	-4.29±0.20	-3.00±0.03	-3.07±0.03
energy	768	8	-0.93±0.01	-2.39±0.01	-0.62±0.03	-1.16±0.03	-0.76±0.10	**-0.57±0.04**	-0.58±0.03
power	9568	4	-2.70±0.00	**-2.67±0.00**	-2.78±0.01	-2.83±0.01	-2.78±0.01	-2.79±0.01	-2.78±0.01
wine	1588	11	-0.91±0.02	**-0.41±0.01**	-1.55±0.05	-0.98±0.01	-1.01±0.02	-0.97±0.01	-0.95±0.02
yacht	308	6	-1.62±0.01	-2.90±0.02	-0.95±0.18	-2.26±0.02	-1.14±0.20	-1.00±0.10	**-0.70±0.08**

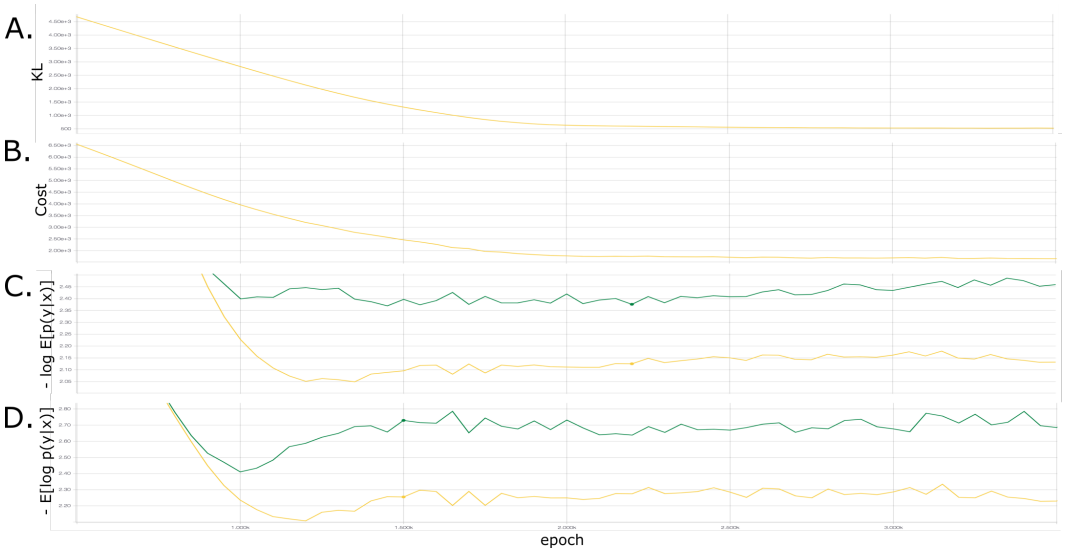


Figure 4: Optimization of variational Bayesian neural networks with mean field approximate posteriors. Weight variances are initialized to be very small which leads to a large initial complexity penalty A.) the complexity penalty or 'KL' term slowly decreases and converges. B.) The overall variational free energy or 'cost' slowly decreases and converges. C.) the expected log likelihood and D.) The expected negative log likelihoods of both the test and training data under the posterior predictive distributions decrease initially but then increase again as the optimization proceeds and the model over-prunes. In C and D, we additionally provide log likelihoods for the test set in green.

A Experimental Details

We use a single hidden layer MLP with 50 hidden units and tanh activations. Our priors and approximate posteriors are diagonal Gaussian. We use a unit normal prior on both weights and biases. We initialize the posterior uncertainties to be 10^{-4}. We train using Adam with parameters $\beta^1 = 0.9$ and $\beta^2 = 0.99$ with a learning rate of 0.005. We ran our optimization for 5000 iterations of batch optimization, except for ES, for which we ran for 2000 epochs (we did not use a validation set, but instead optimized the learning rate on a held out dataset).

We use these the same 20 train-test splits as these previous methods. We optimize hyper-parameters on one of the train/test splits by Bayesian optimization using Spearmint [6][15]). For ES, we optimize the learning rate. For all other models we optimized the prior standard deviation. For GD, we additionally optimize the standard deviation of the approximate posterior. Following [2], we initialize the variances on weights to be very small ($\sigma_{\text{init}} = 10^{-4}$), a trick which empirically seems to provide better results.

[6]Spearmint is openly available at https://github.com/HIPS/Spearmint

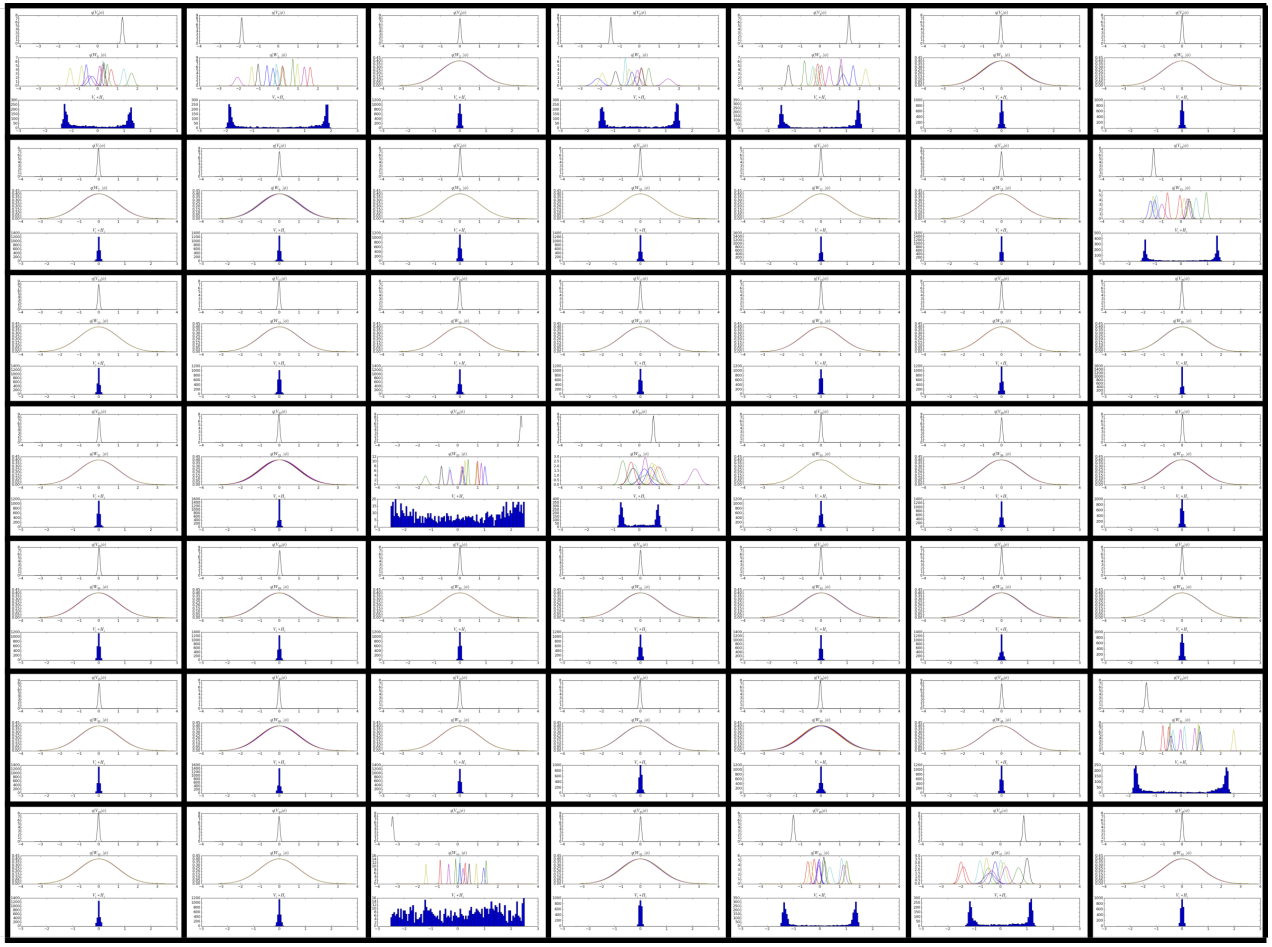


Figure 5: Histograms depicting characteristics of each of the hidden units in of the neural network described in figure 2. Top.) the approximate posterior over the weight connecting it to the output. Middle.) posteriors over weights connecting from the input to the hidden unit Bottom.) Histogram of 25 sampled activations across all training data-points. All but 11 of the hidden units have been pruned. The pruned unit provided in the earlier figure (unit 45) is not included.