Bayesian Learning of Neural Network Architectures

Georgi Dikov **Justin Bayer**
AI Research, Volkswagen Group, Munich, Germany

Abstract

In this paper we propose a Bayesian method for estimating architectural parameters of neural networks, namely layer size and network depth. We do this by learning concrete distributions over these parameters. Our results show that regular networks with a learnt structure can generalise better on small datasets, while fully stochastic networks can be more robust to parameter initialisation. The proposed method relies on standard neural variational learning and, unlike randomised architecture search, does not require a retraining of the model, thus keeping the computational overhead at minimum.

1 INTRODUCTION

One of the reasons for the success of modern deep learning models is attributed to the development of powerful architectures that exploit certain regularities in the data (e.g., convolutional networks such as [Simonyan and Zisserman, 2014, Szegedy et al., 2015]) and alleviate issues with numerical optimisation (e.g., learning an identity mapping in very deep networks [He et al., 2016]). In fact, it has been shown [Saxe et al., 2011] that architecture alone can improve representation learning even with randomly initialised weights.

Traditionally, the architecture of a neural network is treated as a set of static hyperparameters, which are tuned based on an observed performance on a held-out validation set. This viewpoint, however, requires that a network is initialised, trained until convergence and evaluated at each modification of the architecture—a time-consuming procedure which does not allow for an efficient, exhaustive hyperparameter search.

In this work, we propose a scalable Bayesian method to structure optimisation by treating hyperparameters, such as the layer size and network depth, as random variables whose parameterised distributions are learnt together with the rest of the network weights. Taking a Bayesian probabilistic approach to architecture learning is good for two main reasons: (i) the posterior distribution over the architectural parameters reveals whether or not the model has the capacity to represent the training data well; and (ii) imposing prior beliefs over the parameters naturally allows for expert knowledge to be incorporated into the model, without imposing any unbreakable constraints as a side effect. However, obtaining the correct posterior distribution in closed form is not possible due to the highly nonlinear nature of deep neural networks; also residing to a Markov Chain Monte Carlo sampling technique is computationally prohibitive. Instead, we apply the framework of approximate variational inference in order to estimate a posterior distribution over the architectural variables and maintain the differentiability of the model by the means of a continuous relaxation on the discrete categorical (concrete) distribution [Maddison et al., 2016, Jang et al., 2016]. Thus we are able to efficiently evaluate a continuum of architectures. We will show empirically that ensembling predictions from networks of sampled architectures acts as a regulariser and mitigates overfitting.

In the next section we review the necessary background in approximate variational inference, present our model from a Bayesian viewpoint and briefly introduce the concrete categorical distribution. In Section 3 we show the mechanism of layer size and network depth learning and give an intuitive interpretation of the approach. Section 4 compares our method to existing ones and discusses their shortcomings. In Section 5 we evaluate multiple models in regression, classification and bandits tasks and finally we discuss potential consequences in Section 6.

Proceedings of the 22$^{\text{nd}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

2 BACKGROUND AND MODEL STATEMENT

2.1 Approximate Variational Inference

Let $\mathbf{W} = \{\mathbf{W}^1, \mathbf{W}^2, \ldots, \mathbf{W}^n\}$ denote the weights of an n-layer network and $\boldsymbol{\alpha}$ the architectural parameters which are going to be learnt. Further, let $(\mathbf{X}, \mathbf{Y})$ be a labelled dataset. Then, in the framework of Bayesian reasoning, we define a prior distribution $p(\mathbf{W}, \boldsymbol{\alpha}) = p(\mathbf{W})p(\boldsymbol{\alpha})$, a likelihood model $p(\mathbf{Y} \mid \mathbf{X}, \mathbf{W}, \boldsymbol{\alpha})$ and we seek to infer the posterior distribution $p(\mathbf{W}, \boldsymbol{\alpha} \mid \mathbf{X}, \mathbf{Y})$. The latter, however, cannot be evaluated precisely due to the intractability of the normalisation constant $p(\mathbf{Y} \mid \mathbf{X})$. The variational Bayes approach reframes the problem of inferring the posterior distribution into an optimisation one, by minimising an approximation error between a parameterised surrogate distribution $q(\mathbf{W}, \boldsymbol{\alpha} \mid \mathbf{X}, \mathbf{Y})$ and the posterior distribution. For the sake of computational simplicity, throughout this work we will assume that the approximate posterior is fully factorisable, i. e.:

$$q(\mathbf{W}, \boldsymbol{\alpha} \mid \mathbf{X}, \mathbf{Y}) = \prod_{l=1}^{n} q\big(\mathbf{W}^l \mid \mathbf{X}, \mathbf{Y}\big) \prod_{\alpha \in \boldsymbol{\alpha}} q(\alpha \mid \mathbf{X}, \mathbf{Y}) \tag{1}$$

and that the network weights in each layer l, $\mathbf{W}^l$, are independent and Gaussian distributed with parameters $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{|\mathbf{W}^l|}$, i. e. $q\big(\mathbf{W}^l \mid \mathbf{X}, \mathbf{Y}\big) = \mathcal{N}\big(\mathbf{W}^l \mid \boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma}^2)\big)$. Note that relaxing the independence and/or the functional form assumption on the network weights can improve modelling performance, as shown by [Cremer et al., 2018, Pawlowski et al., 2017]. Nevertheless, we leave the extension of architecture learning in Bayesian neural networks with more sophisticated posterior approximation to future work. The prior distribution over the weights $\mathbf{W}^l$ will be a zero-mean factorised Gaussian with the same fixed variance σ_0^2 for each weight, i. e. $p\big(\mathbf{W}^l\big) = \mathcal{N}\big(\mathbf{W}^l \mid \mathbf{0}, \sigma_0^2 I\big)$.

The specific form of $q(\boldsymbol{\alpha})$ and $p(\boldsymbol{\alpha})$ will be elaborated in detail in Section 3 where we will consider learning the layer sizes and the overall network depth. Due to the discrete nature of these parameters, we cannot use backpropagation to learn their posteriors. We will show in Sections 2.2 and 2.3 how we could circumvent this issue.

Let $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ represent the sets of variational parameters for the approximate marginals $q_{\boldsymbol{\eta}}(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})$ and $q_{\boldsymbol{\theta}}(\boldsymbol{\alpha} \mid \mathbf{X}, \mathbf{Y})$ which we denote as $q_{\boldsymbol{\eta}}(\mathbf{W})$ and $q_{\boldsymbol{\theta}}(\boldsymbol{\alpha})$ respectively. One way to quantify the approximation error between the surrogate q and the true posterior p is to measure their Kullback-Leibler divergence [Kullback and Leibler, 1951]. It can be shown that the following relation holds [Jordan et al., 1999]:

$$\boldsymbol{\eta}^*, \boldsymbol{\theta}^* = \underset{\boldsymbol{\eta}, \boldsymbol{\theta}}{\arg\min}\, \mathrm{KL}(q_{\boldsymbol{\eta}}(\mathbf{W})q_{\boldsymbol{\theta}}(\boldsymbol{\alpha}) \;||\; p(\mathbf{W}, \boldsymbol{\alpha} \mid \mathbf{X}, \mathbf{Y})) \tag{2}$$

$$\begin{aligned} &= \underset{\boldsymbol{\eta}, \boldsymbol{\theta}}{\arg\min} - \mathbb{E}_{q_{\boldsymbol{\eta}}(\mathbf{W})q_{\boldsymbol{\theta}}(\boldsymbol{\alpha})}[\log p(\mathbf{Y} \mid \mathbf{X}, \mathbf{W}, \boldsymbol{\alpha})] \\ &\qquad + \mathrm{KL}(q_{\boldsymbol{\eta}}(\mathbf{W}) \;||\; p(\mathbf{W})) \\ &\qquad + \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\alpha}) \;||\; p(\boldsymbol{\alpha})) \end{aligned} \tag{3}$$

$$= \underset{\boldsymbol{\eta}, \boldsymbol{\theta}}{\arg\min} -\mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{\eta}, \boldsymbol{\theta}, \mathbf{X}, \mathbf{Y}). \tag{4}$$

The quantity in Eq. (4), $\mathcal{L}_{\mathrm{ELBO}}$, is called the *Evidence Lower Bound* and will be approximated with Monte Carlo (MC) sampling since the prior, the approximate posterior and the likelihood distributions will have known densities as we will see in Section 3. Also, given that the prior distribution $p(\mathbf{W})$ is a Gaussian, the KL-divergence term for the network weights will be computed analytically and thus will reduce the variance in the gradient estimates. However, the KL-divergence for the architectural parameters $\boldsymbol{\alpha}$ will be estimated using MC sampling. Finally, using the approximations $q_{\boldsymbol{\eta}}(\mathbf{W})$ and $q_{\boldsymbol{\theta}}(\boldsymbol{\alpha})$ we can define a posterior predictive distribution over the labels $\mathbf{Y}$ and approximate it with MC sampling:

$$p(\mathbf{Y} \mid \mathbf{X}) = \iint p(\mathbf{Y} \mid \mathbf{X}, \mathbf{W}, \boldsymbol{\alpha}) q_{\boldsymbol{\eta}}(\mathbf{W}) q_{\boldsymbol{\theta}}(\boldsymbol{\alpha}) d\mathbf{W} d\boldsymbol{\alpha}. \tag{5}$$

Note that even if we treat the network weights $\mathbf{W}$ as point estimates we can still compute an approximate posterior distribution over $\boldsymbol{\alpha}$ and optimise it using the ELBO objective while performing a MAP estimate over $\mathbf{W}$. That is, the approach of Bayesian architecture learning is applicable to regular neural networks as well and we will show such an example in Section 5.

2.2 The Reparameterisation Trick

The reparameterisation trick [Kingma and Welling, 2013] refers to a technique of representing sampling from a probability distribution as a deterministic operation over the distributional parameters and an external source of independent noise. In the context of architecture learning we would like to show that such a reparameterisation is possible for the architectural random variable $\boldsymbol{\alpha}$ of some $\boldsymbol{\theta}$-parameterised distribution $\boldsymbol{\alpha} \sim q_{\boldsymbol{\theta}}(\boldsymbol{\alpha})$. Then, if there is a deterministic and differentiable function g such that $\boldsymbol{\alpha} = g(\boldsymbol{\theta}, \boldsymbol{\epsilon})$ with $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ guaranteeing that $\mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\alpha})}[\boldsymbol{\alpha}] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[g(\boldsymbol{\theta}, \boldsymbol{\epsilon})]$, we can compute the gradient w. r. t. $\boldsymbol{\theta}$ on g and use standard backpropagation to learn $\boldsymbol{\theta}$.

2.3 The Concrete Categorical Distribution

Proposed by [Jang et al., 2016, Maddison et al., 2016] the Gumbel-softmax or concrete categorical distribution is a continuous extension of its discrete counterpart. It is fully reparameterisable as sampling K-dimensional probability vectors $\mathbf{s} \in \Delta^{K-1}$ can be expressed as a deterministic function of its parameters—the probability vector $\boldsymbol{\pi}$—and an external source of randomness $\boldsymbol{\epsilon}$ which is Gumbel-distributed:

$$\begin{aligned} s_i &= \frac{\exp((\log \pi_i + \epsilon_i)/\tau)}{\sum_j \exp((\log \pi_j + \epsilon_j)/\tau)}, \\ \epsilon_i &\sim -\log(-\log\big(\mathrm{Uniform}(0,1)\big)\big). \end{aligned}$$

Here τ is a temperature hyperparameter controlling the smoothness of the approximation. For $\tau \to 0$ the samples become one-hot vectors and for $\tau \to \infty$: $s_i = s_j, \forall i, j$. In this work we will consider τ fixed. The density of the concrete categorical distribution is

$$p(\mathbf{s} \mid \boldsymbol{\pi}, \tau) = (K-1)!\tau^{K-1} \frac{\prod_{i=1}^{K} \pi_i s_i^{-\tau-1}}{\left(\sum_{i=1}^{K} \pi_i s_i^{-\tau}\right)^K}. \tag{6}$$

Analogously for the binary case ($s \in [0,1]$), one can express a sample from a concrete Bernoulli distribution by perturbing the logit with noise from a Logistic distribution and squashing it through a sigmoid:

$$\begin{aligned} s &= \frac{1}{1 + \exp(-(\log(\pi) - \log(1-\pi) + \epsilon)/\tau)}, \\ \epsilon &\sim \mathrm{Logistic}(0,1). \end{aligned}$$

The functional form of its density function is given as:

$$p(s \mid \pi, \tau) = \frac{\tau \pi s^{-\tau-1}(1-s)^{-\tau-1}}{(\pi s^{-\tau} + (1-s)^{-\tau})^2}. \tag{7}$$

For more properties of the concrete distributions see the appendices in [Jang et al., 2016, Maddison et al., 2016].

3 ADAPTIVE NETWORK ARCHITECTURE

In this work we will focus on two important architectural hyperparameters but analogous extensions to others are possible. First we will look into learning the size of an arbitrary layer l denoted with $\mathbf{s}^l$ and then we will proceed with estimating the optimal depth of a network by means of independent layer-wise skip connections γ^l. Following the independence assumption from Eq. (1) for a network of n layers we have:

$$q(\boldsymbol{\alpha}) = \prod_{l=1}^{n} q\big(\mathbf{s}^l\big) q\big(\gamma^l\big). \tag{8}$$

Analogous factorisation applies for the prior $p(\boldsymbol{\alpha})$ as well. In our work, it has the same functional form as the approximate posterior but has fixed parameters.

3.1 Layer Size

Let $\mathbf{s}^l \in \Delta^{K-1}$ be a concrete-categorically distributed random variable encoding the size of an arbitrary fully-connected layer l with maximum capacity of K units[1]. Then the integer number representing the layer size encoded in a sample is given as $k = \arg\max_i s_i^l$. In order to enforce the sampled size on the layer, we propose building a soft and differentiable mask $m(\mathbf{s}^l) \in \Delta^{K-1}$ which multiplicatively gates the output of l:

$$\mathbf{y}^l = f(\mathbf{W}^l \mathbf{y}^{l-1}) \odot m(\mathbf{s}^l) \tag{9}$$

where we omit the bias $\mathbf{b}^l$ for the sake of notational brevity and use f to denote the activation function. Due to the fully-connected nature of the layer, there is in general no preference for which k units should be used. However, one has to be consistent in selecting them across different gradient updates, as this subset of units will represent the reduced in size layer and all others should be discarded, e.g. by deleting $K-k$ rows of $\mathbf{W}^l$. To do this, we construct the mask such that the top k rows are approximately 1s (letting through gradient updates) and the rest 0s (blocking gradient updates). E.g., $m(\mathbf{s}^l) = \mathbf{U}\mathbf{s}^l$ where $\mathbf{U} \in \{0,1\}^{K \times K}$ is an upper triangular matrix of ones. Since $\mathbf{s}^l$ will never be a one-hot vector in practice, the resulting mask will be soft. Note that in a fully Bayesian neural network, the approximate posterior on the parameters of all redundant (blocked) units will conform to the prior, essentially paying a portion of the divergence debt borrowed by the active units.

Before giving explicitly the form of the approximate posterior $q\big(\mathbf{s}^l\big)$ we argue that (i) the learnt distribution should be unimodal, such that a unique optimal layer size can be deduced, and (ii) it should provide us with a meaningful uncertainty estimate. As the probabilities of the concrete categorical distribution are not constrained to express unimodality, we suggest to limit the degrees of freedom by coupling $\boldsymbol{\pi}_i$ through a deterministic and differentiable function. One such candidate is the renormalised density of the truncated Normal distribution which we denote as $\bar{\mathcal{N}}\big(\mu, \sigma^2, 1, K\big)$. By abuse of notation we express $\boldsymbol{\pi}$ as a function of μ and σ and evaluate it at points $\{1, 2, \ldots, K\}$:

$$\boldsymbol{\pi}(\mu, \sigma)_i = \frac{\bar{\mathcal{N}}\big(i \mid \mu, \sigma^2, 1, K\big)}{\sum_{j=1}^{K} \bar{\mathcal{N}}(j \mid \mu, \sigma^2, 1, K)} \quad \text{for } i \in [K], \tag{10}$$

$$q_{\mu,\sigma}\big(\mathbf{s}^l\big) = \mathrm{ConcreteCategorical}(\boldsymbol{\pi}(\mu, \sigma)). \tag{11}$$

[1] Or K filters if the layer is convolutional.

Besides the unimodality, this parameterisation is also advantageous for requiring a constant number of variational parameters w. r. t. the layer size. Throughout this work, the prior $p_{\mu_0,\sigma_0}(\mathbf{s}^l)$ assumes the same parameterisation as $q_{\mu,\sigma}(\mathbf{s}^l)$ and μ_0 and σ_0 are specified in advance. Care must be taken, however, when setting the temperature τ. Since the gradient is scaled with the inverse of τ, small values, e.g. in the order of 0.01, can lead to optimisation instability. We have observed a good performance with a constant temperature in the range of 1.0 to 3.0, which we found empirically. Finally, we note that the gradients w. r. t. the weights and biases are multiplicatively stretched by the sampled mask vector. Therefore, our method can be interpreted as an auxiliary per-unit learning rate, modulating the error signal coming from the data log-likelihood term in the ELBO objective.

3.2 Network Depth

Inspired by [He et al., 2016], we infer the optimal depth of a feed-forward neural network by learning a bypass variable γ^l for each layer independently. Using the notation from above, we can express the layer output $\mathbf{y}^l$ as

$$\mathbf{y}^l = (1-\gamma^l) f(\mathbf{W}^l \mathbf{y}^{l-1}) + \gamma^l \mathbf{y}^{l-1}. \tag{12}$$

We treat γ^l in a Bayesian manner and assume a concrete Bernoulli distribution for the form of the approximate posterior. Thus we learn a single variational parameter π per layer and, again, keep the temperature hyperparameter τ fixed:

$$q_{\pi}(\gamma^l) = \text{ConcreteBernoulli}(\gamma^l). \tag{13}$$

We set the prior $p_{\pi_0}(\gamma^l)$ to be another concrete Bernoulli distribution with fixed parameter π_0. Similarly to the concrete categorical distribution, the temperature hyperparameter τ cannot be small enough so that the sampled bypass coefficient γ^l becomes a numerical 1 or 0. Therefore, in the process of training, the outputs of the skipped layer are only strongly inhibited and not completely shut off but as we will see, this still allows to detect an optimal layer count.

One drawback of the presented approach is its limited applicability to those layers only which do not change the dimensionality of their inputs. The reason is that the skip connection is implemented as a simple convex combination of the layer's input and output as given in Eq. (12). Nevertheless, this method can be used in parallel with the adaptive layer size and thus enable intermediate dimensionality fluctuations. Analogously to the per-unit learning rate argument, we can view the skip connection as a modulation on the gradients to all units and we interpret this method as an adaptive per-layer learning rate.

4 RELATED WORK

Neural network architecture search has long been a topic of research and diverse methods such as evolutionary algorithms [Todd, 1988, Miller et al., 1989, Kitano, 1990], reinforcement learning [Zoph and Le, 2016] or Bayesian optimisation [Bergstra et al., 2013, Mendoza et al., 2016] have been applied. Despite the underlying differences, all these approaches share a common trait in the fact that they decouple the architecture design from the training. Consequently, this has a significant computational burden and to the best of our knowledge, we are the first to oppose to this paradigm and merge weight and architectural hyperparameter optimisation using the forward- and backpropagation cycle of neural network training.

In [LeCun et al., 1990, Hassibi and Stork, 1993] unimportant weights are identified and removed from the architecture. A major limitation is that the initial network architecture can only be reduced. Our approach is similar in the sense that it has an upper limit on the network size, but it also allows for growth after initial contraction, should there be new evidence supporting it. Furthermore the method presented in this work is principled in the inclusion of expert knowledge in the form of fixed prior probability for each layer and only requires the manual tuning of the temperature constant τ.

5 EXPERIMENTS

5.1 Regression on Toy Data

Point-estimate Weights In this first toy data experiment we demonstrate learning a suitable layer size in a single-layer neural network with 50 units and ReLU activation functions. We set a very conservative prior on the size variable $p_{\mu_0,\sigma_0}(\mathbf{s})$ with $\mu_0 = 1$ and $\sigma_0 = 2$ and record the change in the approximate posterior over time. Figure 1 depicts qualitatively the probabilities of the concrete categorical distribution and three snapshots show the current fit over the dataset.

In this example, we generate 2000 points from a one-dimensional noisy periodic function. Due to the large number of data points, the total loss is largely dominated by the data likelihood term and the increasing divergence between the approximate posterior and the prior is acting as a weak regulariser. Consequently, the allocation of more units stops after the data is well approximated. Note that this would not happen, should the prior parameter μ_0 be set to a large value, e.g. 40, as there is no incentive for the model to converge to a

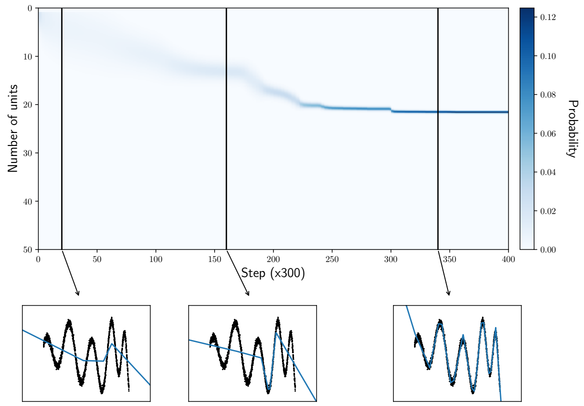


Figure 1: Change in the posterior probabilities $\boldsymbol{\pi}$ over time (as used in Eqs. (10) and (11)). Below the diagram, three snapshots show the fit of the training data: the more units are released, the better the network is able to account for the non-linearity of the data. The optimisation converges to parameters $\mu = 21.99$ and $\sigma = 0.16$. The temperature hyperparameter τ is set to 3.0.

simpler solution. We will see in short that this is no longer the case once we treat the network weights $\mathbf{W}$ in a Bayesian way as well.

Next, we initialise a deep neural network with 11 layers, 10 of which are subject to the bypassing mechanism. In order to enforce the usage of more than one layer we limit the size of each to 5 units and we use again a ReLU activation function. Figure 2 shows the change in the probability of skipping a layer over time. The posterior allows for a clear interpretation that a rigid network of 5 layers will be able to reliably fit the data.

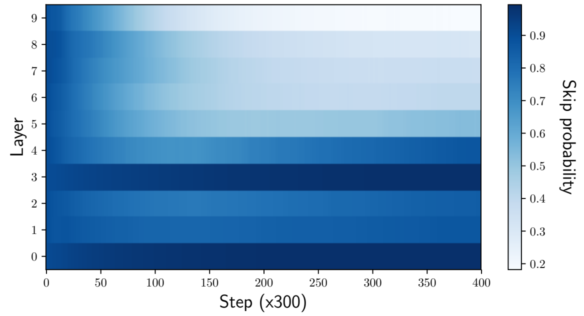
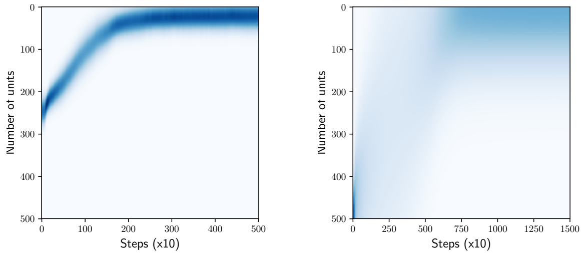


Figure 2: Change in the posterior probabilities $\{\pi^1, \ldots, \pi^{10}\}$ for the skip variables $\{\gamma^l\}_{l\in\{1,2,\ldots,10\}}$ (see Eq. (13)) over time. Five of the layers are bypassed with high probability, indicating that a network with 5 hidden layers of 5 units each is enough to fit the data. The temperature hyperparameter τ is set to 1.0 for each layer.

Bayesian Weights We now construct a fully Bayesian neural network with independently normally distributed weights and biases. In Bayesian neural networks the KL-divergence between the approximate posterior and the prior is acting as a strong regulariser on the parameters and in cases of small data size and overly parameterised models, the noise in the parameters dominates. The aim of this experiment is to show that the presented framework of architecture optimisation mitigates this issue by not only extending inadequately small architectures but also reducing oversized ones. Figures 3a and 3b show the change in posterior for two different priors: one with $\mu_0 = 250$ and $\sigma_0 = 20$ and another with $\mu_0 = 500$ and $\sigma_0 = 50$. Notice that in both cases the variational parameter μ converges to approximately the same value, suggesting that the method is robust to setting inappropriate prior distributions.



(a)

(b)

Figure 3: Change in posterior over the size of a single-layer Bayesian neural network. Prior parameters: (a) $\mu_0 = 250$ and $\sigma_0 = 20$ and (b) $\mu_0 = 500$ and $\sigma_0 = 50$. The temperature hyperparameter τ is set to 3.0.

In addition, we performed experiments where the layer size and the network depth are jointly learnt. In the cases where the architectural prior is on very few units and layers, as in Figure 1, the network first allocates more layers. This is an easier way to increase capacity in comparison to adding more units to a layer. It has, however, one important consequence—having a very deep but narrow Bayesian neural network can be computationally inconvenient, as the variance in the output becomes intractably large. One way to alleviate this problem would be to balance the network depth and layer size, e.g. by choosing an appropriate prior connecting the size and skip variables. We leave this to future research.

5.2 Regression on UCI Datasets

We explored the robustness in performance of Bayesian neural networks on several real-world datasets [Dheeru and Karra Taniskidou, 2017]. We trained shallow and deep rigid networks and their architecture-regularised counterparts for 200 epochs with small batch size of 8. The shallow model comprises of a single ReLU-activated layer with 50 units

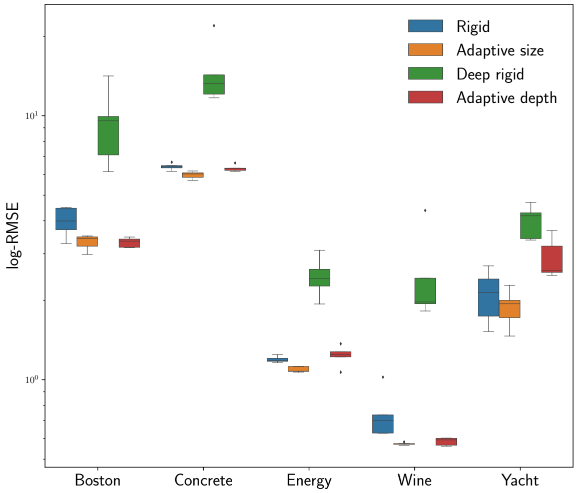


Figure 4: Test set RMSE performance on 5 UCI datasets for single-layer rigid and adaptive and deep rigid and adaptive Bayesian neural networks. Lower is better.

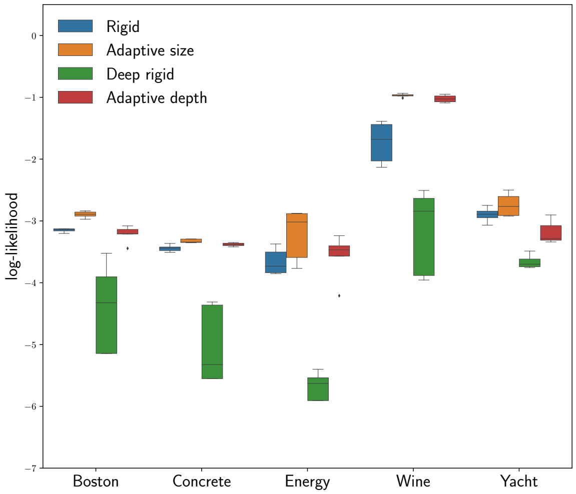


Figure 5: Test set log-likelihood performance on 5 UCI datasets for single-layer rigid and adaptive and deep rigid and adaptive Bayesian neural networks. Higher is better.

and the deep one stacks 5 of them. In all cases the prior distributions over the structural variables were initialised with parameters $\mu_0 = 25$, $\sigma_0 = 10$ for the size mechanism, $\pi_0 = 0.1$ for the layer bypassing one. All network weights have a standard normal prior. The posterior approximation over the weights is initialised from the prior as well. As in the previous experiments, the temperature parameters τ are kept fixed at 3.0 and 1.0 for the layer size and network depth respectively. The datasets chosen for this experiment are multidimensional (varying between 6 and 13 features) and contain a fairly small amount of samples (between 300 and 1500), which results in very noisy predictions on the overparameterised models.

We show that learning the structure has significant benefits in performance measured as a root mean squared error (RMSE) and log-likelihood on a held-out test set. The experiments have been repeated 20 times. In Fig. 4 the RMSE of the depth and size adaptive models are lower meaning that they generalise better and the standard deviations narrower, signifying a robustness to initialisations. The results for the log-likelihood in Fig. 5 show that the structure-regularised models are less uncertain about the predictions. Deep rigid models however, fail to fit the data as the noise in the network weights is prevailing. Moreover, both rigid models are highly dependent of the particular parameter initialisation, which is reflected in the large standard deviations in the box plots. On the other hand, the performance of the adaptive models is consistent throughout independent experiment repetitions.

5.3 Contextual Bandits

In this experiment we set up a discrete decision making task where an agent's action $a \in \mathcal{A}$ triggers a reward $r \in \mathbb{R}$ from the environment, i.e. the bandit. At each time step the agent's action is conditioned on a context $c \in \mathcal{C}$ which is independent of all previous ones. Hereby we aim to show the versatility of the adaptive architecture approach in an online learning scenario as changing the quality and quantity of the data changes the requirements for a network structure.

In the bandit task the goal of the agent is to maximise the expected received reward, or equivalently, to minimise the expected regret. The latter is defined as the difference in the rewards received by an oracle and the agent. In order to perform optimally, the agent learns an approximation $f(a,c) : (\mathcal{A} \times \mathcal{C}) \to \mathbb{R}$ to the bandit's intrinsic reward function and uses it to pick an action. The current context, performed action and received reward are then kept in a data buffer.

The reward approximation function f is parameterised as a Bayesian neural network with weights $\mathbf{W}$ and a prior $p(\mathbf{W})$. Furthermore, let $p(r \mid a,\, c,\, \mathbf{W})$ be the likelihood of a reward r under $f_{\mathbf{W}}$. Then, using variational inference we can define a Bayesian objective and learn an approximate posterior $q_{\boldsymbol{\theta}}(\mathbf{W})$. Using the likelihood term $p(r \mid a,\, c,\, \mathbf{W})$, we can now define the optimal action as the one that maximises the expected reward. After performing the action we then update $q_{\boldsymbol{\theta}}(\mathbf{W})$ and repeat for the next context sample. This iterative approach is called Thompson sampling [Thompson, 1933] and was developed as an efficient way to tradeoff exploration for exploitation in the framework of Bayesian decision making.

In the following we compare agents with purely greedy, randomised and (adaptive) Bayesian reward estimation models. The purely greedy agent is deterministic in nature and always picks the action with highest reward estimate for a given context. The randomised or ϵ-greedy agent performs the estimated best action with probability $1 - \epsilon$, otherwise a random one is chosen. This way, despite the agent's deterministic reward model it will still explore potentially better options. Nevertheless, if ϵ is not annealed during the interaction with the bandit, the agent will never achieve a 0 expected regret, even with a perfect reward model. The Bayesian agent, however, will explore more actively in the beginning when few data are seen, and will transition automatically into an exploitation regime once the uncertainty in the posterior becomes small enough. The speed at which this transition happens depends on the prior, the initialisation and the variance in the gradients.

Following [Blundell et al., 2015] we evaluate the agents on the Mushroom UCI dataset [Dheeru and Karra Taniskidou, 2017] consisting of more than 8000 mushrooms, described as categorical vectors of features. T he task is then to decide whether or not to consume a given mushroom. If it is labelled as poisonous and is being consumed the agent receives a randomised reward of either -35 or 5 with 50% chance each. If the consumed mushroom is edible the reward is positive 5. All rejected samples receive a reward of 0. In this experiment we measure the cumulative regret over the course of 30 000 interactions. Both the greedy and Bayesian agents are parameterised by 2-layer neural networks with 100 units and ReLU activations in each layer. The adaptive Bayesian agent has a prior centred at 50 units and a broad standard deviation of 20. For the sake of computational efficiency, we do not retrain the reward model at each new bandit interaction but only fine-tune it with one epoch on the current dataset buffer whose size is limited to the last 4096 samples. We used a learning rate of 0.0005 and initialised the standard deviations of the Bayesian weights at 0.02. The reported results are the average of 5 independent runs of the experiment.

Throughout the experiments, the Bayesian rigid agent consistently encountered stability issues and after about 20 000 interactions the reward estimates became so unreliable, that the model settled for the suboptimal solution of picking the *reject* action for all observed mushrooms. Fig. 6 shows the cumulative regret over time. The failure of the rigid Bayesian model is due to a numerical instability arising from huge gradients caused by wrong reward guesses as it can be seen in the plot of the reward RMSE in Fig. 7. Clearly, the suboptimal behaviour of the Bayesian rigid agent is remedied by the adaptive size regularisation.

In addition, we show the benefits of the learnt architecture by initialising a new one from the converged posterior approximation over the size, in this case—two layers with 34 and 20 units accordingly. It has best performance among the Bayesian and greedy agents with the only exception being the purely greedy agent. We attribute its surprising success to chance and claim without proof that a more challenging dataset will be able to display its lack of principled exploration skills.

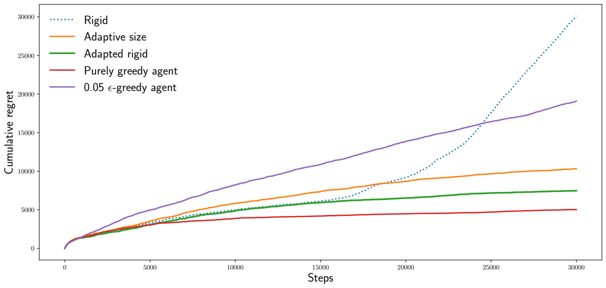


Figure 6: Cumulative regret, aggregated over 30 000 randomly presented context vectors. The estimated reward is modelled by 2-layer rigid and adaptive size Bayesian neural networks. The rigid network consistently exhibits instability after about 17 000 steps, while the adaptive one remains stable. The best performance among all Bayesian models is obtained by a rigid network whose architecture is initialised from the converged structural parameters of the adaptive network. As a baseline $0.05 - \epsilon$ and purely greedy agents are evaluated.

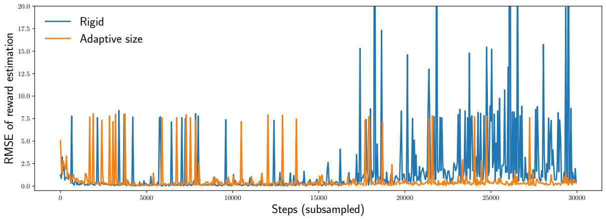


Figure 7: Reward RMSE for the rigid and adaptive agents. The instability in the estimate results in suboptimal behaviour in action picking and hence a substantial increase in cumulative regret.

5.4 Image Classification

To demonstrate the broad applicability of the proposed adaptive architecture method, we apply it on the filter count hyperparameter in Bayesian convolutional neural networks. The extension from the fully connected layers to the output channels of a convolutional layer is straightforward. Similarly, the adaptive network depth regularisation remains unchanged. In this case though, the number of channels from the previous layer should match the one from the current. All experiments are performed on three popular 10-class datasets of increasing discrimination difficulty: MNIST [LeCun et al., 2010], Fashion MNIST [Xiao et al., 2017] and CIFAR-10 [Krizhevsky et al., 2014]. The training sets of these are comprised of 60 000, 60 000 and 50 000 samples respectively and all results presented are based on the average of 100 samples form the model predictive distribution over the held-out 10 000 test samples.

We check the advantage of the adaptive size regularisation in a fairly "wide" model architecture consisting of three Bayesian convolutional layers, each followed by a ReLU non-linearity and a max pooling operation and two Bayesian fully-connected layers. The first two layers have a window size of 5 and the third of 3. The layers host 81, 64, and 64 filters respectively and padding is added to preserve the input dimensionality. After the convolutional layers, the data is flattened and processed by a ReLU-activated fully-connected layer of size 64 and fed into a softmax output layer. For the adaptive network we apply the size regularisation after each convolutional layer. The priors over the size parameters are set to 80% of the maximum filter count and we set $\tau = 3.0$. All configurations are trained for 200 epochs using early stopping, the Bayesian layers have a standard normal prior and the standard deviations of the network weights are initialised to 0.05. Additionally, we create a deep architecture with 9 convolutional layers grouped into 3 blocks of 3 consecutive layers with 32 filters (16 for the first block only) and a max-pooling operation at the end. For the adaptive depth networks, the second and third layer in each block are skipped. We set a very conservative skip prior probability $\pi_0 = 0.1$ and keep the temperature constant at $\tau = 1.0$. At the end of the third block, the data is flattened and passed through the fully-connected ReLU and softmax output layers as described above. All other training configurations remain the same.

We evaluate all four neural network configurations in two experimental scenarios. In the first one we learn the parameters from the full training dataset and in the second we reduce each to 1000 randomly chosen samples. Table 1 shows the test set accuracy on the full dataset size (top) and on the reduced one (bottom) for the Bayesian models. There is a clear advantage of the adaptive networks over the rigid ones and it is only amplified by the difficulty of the dataset—the improvement in test set accuracy on the reduced CIFAR-10 is almost 4%. We remark, however, that even the best of these results are not representative for the state-of-the-art and that the purpose of the experiment is to compare the influence of the adaptive architecture method in a rather generic setup.

Dataset	Rigid	Adaptive size	Deep rigid	Adaptive depth
MNIST	99.34	**99.40**	**99.46**	99.42
Fashion	**91.41**	91.13	91.14	**91.22**
CIFAR-10	73.31	**74.06**	68.51	**69.63**
MNIST	94.47	**95.67**	**95.72**	94.81
Fashion	79.69	**81.18**	80.32	**80.83**
CIFAR-10	34.98	**38.95**	33.83	**37.49**

Table 1: Test set accuracy on the full (top) and reduced (bottom) datasets for "wide" rigid and adaptive as well as "deep" rigid and adaptive Bayesian convolutional neural networks.

6 CONCLUSION

In this work we introduced a novel method for learning a neural network architecture by including discrete hyperparameters such as the layer size and the network depth into the Bayesian framework. We used parameterised concrete distributions over the architectural variables and variational inference to approximate their posterior distributions. T his allowed us to learn the network structure without significant computational overhead, to sweep through a continuous hyperparameter space and to incorporate external knowledge in the form of prior distributions. The interpretability of the approximate posterior distribution over the layer size and network depth parameters gave us a tool to identify architectural misspecifications and choose optimal values for the layer dimensions. We showed empirically the benefits of the methods in predictive tasks on regression and classification datasets where regularised network structures demonstrated superior test set performance.

Acknowledgements

We thank Botond Cseke and Atanas Mirchev for their astute remarks and invaluable advice for improving the quality of this work.

References

[Bergstra et al., 2013] Bergstra, J., Yamins, D., and Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org.

[Cremer et al., 2018] Cremer, C., Li, X., and Duvenaud, D. (2018). Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*.

[Dheeru and Karra Taniskidou, 2017] Dheeru, D. and Karra Taniskidou, E. (2017). UCI machine learning repository.

[Hassibi and Stork, 1993] Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer.

[Jang et al., 2016] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[Jordan et al., 1999] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Kitano, 1990] Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex systems*, 4(4):461–476.

[Krizhevsky et al., 2014] Krizhevsky, A., Nair, V., and Hinton, G. (2014). The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

[LeCun et al., 2010] LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2.

[LeCun et al., 1990] LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

[Maddison et al., 2016] Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

[Mendoza et al., 2016] Mendoza, H., Klein, A., Feurer, M., Springenberg, J. T., and Hutter, F. (2016). Towards automatically-tuned neural networks. In *Workshop on Automatic Machine Learning*, pages 58–65.

[Miller et al., 1989] Miller, G. F., Todd, P. M., and Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In *ICGA*, volume 89, pages 379–384.

[Pawlowski et al., 2017] Pawlowski, N., Rajchl, M., and Glocker, B. (2017). Implicit weight uncertainty in neural networks. *arXiv preprint arXiv:1711.01297*.

[Saxe et al., 2011] Saxe, A. M., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., and Ng, A. Y. (2011). On random weights and unsupervised feature learning. In *ICML*, pages 1089–1096.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al. (2015). Going deeper with convolutions. Cvpr.

[Thompson, 1933] Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

[Todd, 1988] Todd, P. (1988). Evolutionary methods for connectionist architectures. *Psychology Dept. Stanford University, unpublished Manuscript*.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[Zoph and Le, 2016] Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.