Practical Bayesian Neural Networks via Adaptive Subgradient Optimization Methods

Arnold Salas*
University of Oxford
arnold.salas@eng.ox.ac.uk

Samuel Kessler*
University of Oxford
skessler@robots.ox.ac.uk

Stefan Zohren
University of Oxford
zohren@robots.ox.ac.uk

Stephen Roberts
University of Oxford
sjrob@robots.ox.ac.uk

Abstract

We introduce a novel framework for the estimation of the posterior distribution over the weights of a neural network, based on a new probabilistic interpretation of adaptive subgradient algorithms such as ADAGRAD and ADAM. We demonstrate the effectiveness of our Bayesian ADAM method, BADAM, by experimentally showing that the learnt uncertainties correctly relate to the weights' predictive capabilities by weight pruning. We also demonstrate the quality of the derived uncertainty measures by comparing the performance of BADAM to standard methods in a Thompson sampling setting for multi-armed bandits, where good uncertainty measures are required for an agent to balance exploration and exploitation.

1 Introduction

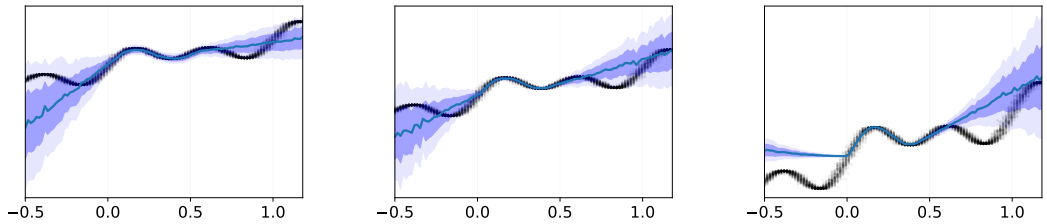
For decades, many different approaches have been suggested to integrate Bayesian inference and neural networks. The principal property of Bayesian neural networks (BNNs) is not a single set of parameters or weights, but an (approximate) posterior distribution over those parameters. The posterior distribution enables uncertainty estimates over the network output, selection of hyperparameters and models in a principled framework, and guided data collection (active learning) for instance.

In general, exact Bayesian inference over the weights of a neural network is intractable as the number of parameters is very large and the functional form of a neural network does not lend itself to exact integration. For this reason, much of the research in this area has been focused on approximation techniques. Most modern techniques stem from key works which used either a Laplace approximation [1], variational methods [2], or Monte Carlo methods [3]. Over the past few years, many methods for approximating the posterior distribution have been suggested, falling into one of these categories. These methods include assumed density filtering [4, 5], approximate power Expectation Propagation [6], Stochastic Langevin Gradient Descent [7–9], incremental moment matching [10] and variational Bayes [11, 12].

The standard variational Bayes approach developed by [11], called Bayes By Backprop (BBB), has several shortcomings. The variational free energy minimized in BBB is a sum of a log-likelihood cost function and a complexity cost function. The complexity cost function acts as a regularizer, enforcing a solution that captures the complexity of the data while keeping the posterior close to the prior. Finding a good prior is usually a non-trivial task, and over-restricting priors could potentially cause underfitting. To alleviate these issues, [13] introduced variational dropout, which uses an

*Equal contribution.

Preprint. Under review.



(a) BADAM (b) MC dropout (c) BBB

Figure 1: Qualitative comparison of the predictive uncertainties of our method and our principle baselines. BNNs are trained on samples from the interval $(0, 0.5)$ and evaluated on the interval $(-0.5, 1.2)$. The plots show the mean prediction and $\pm 1, 2$ standard errors of the predictive distributions.

improper prior to ensure that the complexity cost function becomes constant in the weight parameters. Later modifications of this approach [14–16] were shown to be useful for weight pruning (without re-training, similarly to BBB). However, [17] recently pointed out that such variational dropout approaches are not Bayesian. To avoid these issues altogether, [18] proposed an online variational Bayes scheme for lifelong learning using a new prior for each minibatch, instead of one prior for all the data.

In this paper, we develop a novel Bayesian approach to learning for neural networks, built upon adaptive subgradient methods such as ADAGRAD [19], RMSPROP [20] and ADAM [21]. Unlike the aforementioned approaches, ours does not require the specification of a method for approximating the posterior distribution, as it relies on a new probabilistic interpretation of adaptive subgradient algorithms that effectively shows these can readily be utilized as approximate Bayesian posterior inference schemes. This has similar underpinnings as the work of [22], although the latter is based on a stochastic model for gradient variations that imposes a number of restrictions on the gradient noise covariance structure, which our framework is able to sidestep by utilizing ADAM as the underlying subgradient method. Our proposed algorithm is also similar in spirit to the work of [14]. However [14] performs natural gradient variational inference, implemented within ADAM via weight perturbation of the gradient evaluation.

Our method, BADAM, produces similar predictive distributions to common benchmarks BBB [11] and MC DROPOUT [13] (see Figure 1). We follow the same experimental setup as [4, 11]. The predictive distributions are obtained through sampling from the posterior weight distributions when making predictions on a test set. Experimental details are provided in appendix A.

2 Preliminaries

Notation. Vectors are denoted by lower case Roman letters such as a, and all vectors are assumed to be column vectors. Upper case roman letters, such as M, denote matrices, with the exception of the identity matrix which we denote by $\mathbb{1}$ and whose dimension is implicit from the context. Finally, for any vector $g_i \in \mathbb{R}^d$, $g_{i,j}$ denotes its jth coordinate, where $j \in [d]$.

Problem setup. Let $f(\theta)$ be a noisy objective function, a scalar function that is differentiable w.r.t. the parameters $\theta \in \Theta$, where Θ denotes the parameter space. In general, Θ is a subset of $\mathbb{R}^d$, but for simplicity, we shall assume that $\Theta = \mathbb{R}^d$ throughout the remainder of this paper. We are interested in minimizing the expected value of this function, $\mathbb{E}[f(\theta)]$, w.r.t. its parameters θ. Let $f_1(\theta), \ldots, f_T(\theta)$ denote the realizations of the stochastic function at the subsequent time steps $t \in [T]$. The stochastic nature may arise from the evaluation of the function at random subsamples (minibatches) of datapoints, or from inherent function noise.

The simplest algorithm for this setting is the standard online gradient descent algorithm [23], which moves the current estimate θ_t of θ in the opposite direction of the last observed (sub)gradient value $g_t = \nabla f_t(\theta_t)$, i.e.,

$$\theta_{t+1} = \theta_t - \eta_t g_t, \tag{1}$$

where $\eta_t > 0$ is an adaptive learning rate that is typically set to $\eta/\sqrt{t}$, for some positive constant η. While the decreasing learning rate is required for convergence, such an aggressive decay typically translates into poor empirical performance.

Generic adaptive subgradient descent. We now present a framework that contains a wide range of popular adaptive subgradient methods as special cases, and highlight their flaws and differences. The presentation here follows closely that of [24]. The update rule of this generic class of adaptive methods can be compactly written in the form

$$\theta_{t+1} = \theta_t - \eta_t V_t^{-1/2} m_t, \tag{2}$$

where m_t and $V_t^{-1/2}$ are estimates of the (sub)gradient and inverse Hessian, respectively, of the functions $f_t(\cdot)$, based on observations up to and including iteration t. In other words, they are functions of the (sub)gradient history $g_{1:t} \equiv g_1, \ldots, g_t$, which we express as

$$m_t = \widehat{g}_t(g_{1:t}), \qquad V_t^{1/2} = \widehat{H}_t(g_{1:t}), \tag{3}$$

where $\widehat{g}_t(\cdot)$ and $\widehat{H}_t(\cdot)$ denote approximation functions for the (sub)gradient and Hessian of the loss function at iteration t, respectively. The corresponding procedure is repeated until convergence.

For computational performance many popular algorithms restrict themselves to diagonal variants of adaptive subgradient descent, such that $V_t = \mathrm{diag}(v_t)$, where v_t is the vector of diagonal elements. We first observe that the standard online gradient descent (OGD) algorithm arises as a special case of this framework if we use:

$$m_t = g_t, \qquad V_t = \mathbb{I}. \tag{OGD}$$

The key idea of adaptive methods is to choose estimator functions appropriately so as to entail good convergence. For instance, the first adaptive method AdaGrad [19], which led to considerable usage, employs the following estimator functions:

$$m_t = g_t, \qquad V_t = \frac{1}{t}\mathrm{diag}\left(\sum_{i=1}^{t} g_i g_i^\top\right). \tag{AdaGrad}$$

In contrast to the learning rate of $\eta/\sqrt{t}$ in OGD with learning-rate decay, such a setting effectively implies a modest learning-rate decay of $\eta/\sqrt{\sum_i g_{i,j}^2}$ for $j \in [d]$. When the gradients are sparse, this can potentially lead to huge gains in terms of convergence (see [19]). These gains have also been observed in practice even in some non-sparse settings.

Adaptive methods based on EWMA. Exponentially weighted moving average (EWMA) variants of AdaGrad are popular in the deep learning community. AdaDelta [25], RMSProp [20], Adam [21] and Nadam [26] are some prominent algorithms that fall in this category. The key difference between these and AdaGrad is that they use an EWMA as the function V_t instead of a simple average. Adam, a particularly popular variant, is based on the following estimator functions:

$$m_t = \frac{1-\beta_1}{1-\beta_1^t}\sum_{i=1}^{t}\beta_1^{t-i} g_i, \qquad V_t = \frac{1-\beta_2}{1-\beta_2^t}\mathrm{diag}\left(\sum_{i=1}^{t}\beta_2^{t-i} g_i g_i^\top\right), \tag{Adam}$$

where $\beta_1, \beta_2 \in [0, 1)$ are exponential decay rates. This update can alternatively be stated in terms of the following simple recursions:

$$m_{t,i} = \frac{\beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}}{1-\beta_1^t}, \qquad v_{t,i} = \frac{\beta_2 v_{t-1,i} + (1-\beta_2) g_{t,i}^2}{1-\beta_2^t}, \tag{4}$$

for all $t \in [T]$, with $m_{0,i} = v_{0,i} = 0$ for all $i \in [d]$. Note that the denominator represents a bias-correction term. A value of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ is typically recommended in practice [21]. RMSProp, which appeared in an earlier unpublished work [20], is essentially a variant of Adam with $\beta_1 = 0$. In practice, especially in deep-learning applications, the momentum term arising due to non-zero β_1 appears to significantly boost performance.

Bayesian neural networks. As the name suggests, a Bayesian neural network (BNN) is a neural network equipped with a prior distribution over its weights θ. Consider an i.i.d. data set of N feature vectors $x_1, \ldots, x_N \in \mathbb{R}^d$, with a corresponding set of outputs $\mathcal{D} = \{y_1, \ldots, y_N\} \subset \mathbb{R}$. For illustration purposes, we shall suppose that the likelihood for each datapoint is Gaussian, with an x-dependent mean given by the output $\mathrm{NN}(x, \theta)$ of a neural-network model and with variance σ^2:

$$p(y_n|x_n, \theta, \sigma^2) = \mathcal{N}(y_n|\mathrm{NN}(x_n, \theta),\, \sigma^2). \tag{5}$$

Similarly, we shall choose a prior distribution over the weights θ that is Gaussian of the form

$$p(\theta|\alpha) = \mathcal{N}(\theta|0,\, \alpha^{-1}\mathbb{I}). \tag{6}$$

Since the data set $\mathcal{D}$ is i.i.d., the likelihood function is given by

$$p(\mathcal{D}|\theta, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y_n|\mathrm{NN}(x_n, \theta),\, \sigma^2) \tag{7}$$

and so by virtue of Bayes' theorem, the resulting posterior distribution is then

$$p(\theta|\mathcal{D}, \alpha, \sigma^2) \propto p(\theta|\alpha)p(\mathcal{D}|\theta, \sigma^2) \tag{8}$$

which, as a consequence of the nonlinear dependence of $\mathrm{NN}(x, \theta)$ on θ, will be non-Gaussian. However, we can find a Gaussian approximation by using the Laplace approximation [1]. Alternative approximation methods have been briefly discussed in Section 1.

3 Probabilistic Interpretation of Adaptive Subgradient Methods

We have touched upon the fact that adaptive subgradient methods use estimates of the curvature of the loss function for optimization. Now we will explore how using these estimates one can obtain a posterior around the local minimum which the optimizer converges to, by a Laplace approximation of our likelihood. Consider a second-order Taylor expansion of the loss function $f(\theta)$ around the current parameter estimate θ_t:

$$f(\theta) \approx f(\theta_t) + \nabla f(\theta_t)^\top(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^\top \nabla^2 f(\theta_t)(\theta - \theta_t). \tag{9}$$

Since the gradient and Hessian are unknown, we replace them with the approximations used in adaptive subgradient methods introduced in the previous section

$$\nabla f(\theta_t) \approx \eta_t m_t, \quad \nabla^2 f(\theta_t) \approx V_t^{1/2}, \tag{10}$$

which results in the following approximation:

$$\widehat{f}_t(\theta) \equiv f(\theta_t) + \eta_t m_t^\top(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^\top V_t^{1/2}(\theta - \theta_t). \tag{11}$$

The corresponding likelihood model is given by

$$\ell_t(\theta) \propto \exp\left\{-N\widehat{f}_t(\theta)\right\} \propto \exp\left\{-N\eta_t m_t^\top(\theta - \theta_t) - \frac{N}{2}(\theta - \theta_t)^\top V_t^{1/2}(\theta - \theta_t)\right\}, \tag{12}$$

where the number of samples N corrects the averaging over samples that is implicitly contained in the loss function $f(\theta)$.

Considering a diagonal Gaussian prior over θ, i.e. $p(\theta) = \mathcal{N}(\theta \,|\, 0, \sigma^2)$. Completing the square with respect to θ in the exponential yields

$$\begin{aligned} &- N\eta_t m_t^\top(\theta - \theta_t) - \frac{N}{2}(\theta - \theta_t)^\top V_t^{1/2}(\theta - \theta_t) - \frac{1}{2\sigma^2}\theta^\top\theta \\ &= -\frac{N}{2}(\theta - \theta_{t+1})^\top V_t^{1/2}(\theta - \theta_{t+1}) - \frac{1}{2\sigma^2}\theta^\top\theta + \text{const}, \end{aligned} \tag{13}$$

where, as a reminder, $\theta_{t+1} = \theta_t - \eta_t V_t^{-1/2} m_t$ and "const" denotes terms that are independent of θ. This leads to a Gaussian posterior of the form

$$p(\theta|g_{1:t}, \eta_t, \omega) = \mathcal{N}\left(\theta \,\middle|\, \left(NV_t^{1/2} + 1/\sigma^2\right)^{-1}\left(NV_t^{1/2}\right)\theta_{t+1},\, \left(NV_t^{1/2} + 1/\sigma^2\right)^{-1}\right), \tag{14}$$

where ω is a vector of hyperparameters other than the learning rate, if any, that govern the underlying adaptive subgradient method. For example, $\omega = \emptyset$ in the case of ADAGRAD, whereas $\omega = (\beta_1, \beta_2)$ for ADAM. $V_t^{1/2}$ is used as an approximation for the curvature of the energy landscape and intuitively if the curvature is very high in a given direction then this leads to less variability of the parameters in this direction, on the other hand if the energy landscape is very flat one expects larger variability. The notion of curvature can thus be readily interpreted as a geometric notion of uncertainty.

A few comments are in order regarding Eq. 14.

- The posterior mean is an interesting expression. Consider an eigendecomposition of our Hessian which diagonalizes for interpretational ease. Then, if $V_t^{1/2} = 0$, there is a large uncertainty in the energy landscape and the weight is implicitly pruned. If $NV_t^{1/2} \approx 1$ then the coefficient of our point estimate is less than 1 and we get the traditional interpretation of a Gaussian prior as an L^2 regularizer. And if, $NV_t^{1/2}$ is very large, there is a large curvature in the energy landscape and it is pretty certain that our optimizer has settled on a local minimum; the mean is around the point estimate of our weight. This is also the case for most of the alternative approaches (e.g., [11, 22, 18]).
- The posterior variance is a trade-off between the uncertainty induced by the curvature of our energy landscape and the prior variance. The expression for the variance, and the nature of our approach in general, is closest in spirit to the work of [22]. In fact, Eq. 12 is closely related to Assumption 4 in their paper. A notable difference between their paper and ours, however, is that the former relies on an Ornstein-Uhlenbeck process to describe the stochastic dynamics of the gradients. Specifically, [22] assume that the variability of the gradients can be reasonably captured by a constant covariance matrix, which is an unrealistic assumption given the fact that, in practice, this covariance matrix evolves as one explores different regions of the energy landscape. Instead, our approach, as we shall discuss in the next section, is to use ADAM's EWMA estimates for m_t and V_t. This enables us to filter out the noise arising from the stochastic nature of the gradients, while at the same time accounting for changes in these quantities over different areas of the energy landscape.
- N can be treated as a hyperparameter similarly to [27, 28]. With this in mind we may tune N to be very large and hence the posterior mean of the weight distribution is merely centered on the point estimator of the descent algorithm.

Having introduced a posterior over our weights using estimates from an adaptive subgradient optimizer, we can now encapsulate this into a Bayesian optimization algorithm in the next section.

4 Practical Algorithms for Bayesian Learning of Neural Networks

Based on the insights from the previous section, we obtain the following generic algorithm for Bayesian learning of neural networks via adaptive subgradient methods.

Algorithm 1 Generic Bayesian Learning of Neural Networks via Adaptive Subgradient Methods

Input: $\theta_1 \in \mathbb{R}^d$, learning-rate schedule $\{\eta_t\}_{t=1}^T$, sequence of (sub)gradient and Hessian estimators $\{\widehat{g}_t(\cdot),\, \widehat{H}_t(\cdot)\}_{t=1}^T$
for $t = 1$ **to** $T - 1$ **do**
  $g_t = \nabla f_t(\theta_t)$
  $m_t = \widehat{g}_t(g_{1:t})$ and $V_t^{1/2} = \widehat{H}_t(g_{1:t})$
  $\theta_{t+1} = \theta_t - \eta_t V_t^{-1/2} m_t$
end for
Output: approximate posterior $\mathcal{N}\left(\theta \,\middle|\, \left(NV_t^{1/2} + 1/\sigma^2\right)^{-1} \left(NV_t^{1/2}\right) \theta_{t+1},\, \left(NV_t^{1/2} + 1/\sigma^2\right)^{-1}\right)$

To approximate the correct covariance of the weights, one needs a good estimate of the curvature. Furthermore, due to the stochastic nature of the energy surface, this curvature estimate should not be based on a single (the final) observation, but rather on a history of observations, so as to mitigate

the noise in the resulting curvature estimate. Algorithms like ADAGRAD and ADAM do exactly that. Furthermore, since working with a full covariance matrix becomes computationally intractable for large networks, one can use approximations that diagonalize the covariance matrix. This occurs when using Algorithm 1 with the values taken by the estimator functions $\widehat{g}_t(g_{1:t})$ and $\widehat{H}_t(g_{1:t})$ in ADAGRAD and ADAM. Note that the online variational Bayes algorithms in [11] and [18] also employ a diagonal approximation. Additionally, the model discussed in [22] arguably bears certain similarities with Algorithm 1 when refining the latter to ADAGRAD. We shall focus here on discussing how to instantiate Algorithm 1 with ADAM.

ADAM has many appealing features when used as an optimizer for neural networks. The reasons for this are twofold:

1. Given that the posterior mean of our algorithm is related to the point estimate generated by ADAM, we expect it to perform well in practice, for the same reasons that ADAM excels and is widely used in practice.
2. There is a trade-off in estimating the curvature of the landscape, and thus the covariance matrix: if we just focus on the last observation, our estimate will be too noisy; however, if we base it on the entire history – like ADAGRAD does – we are implicitly assuming that it is constant throughout the landscape. Ideally, therefore, we should base our estimate on the most recent observations close to the final weight update. This is achieved by using EWMAs, as in ADAM[2]. Additionally the use of dropout will add perturbations to loss function landscape which will encourage better curvature estimates in conjunction with an EWMA.

The specific approach whereby Algorithm 1 uses the update rules of ADAM is illustrated in Algorithm 2.

Algorithm 2 BADAM: Bayesian Learning of Neural Networks via ADAM

Input: $\theta_1 \in \mathbb{R}^d$, global learning rate η, exponential decay rates β_1, β_2, constant ϵ
Set $m_0 = v_0 = 0$
for $t = 1$ **to** $T - 1$ **do**
$g_t = \nabla f_t(\theta_t)$
$m_t = \frac{\beta_1 m_{t-1} + (1-\beta_1) g_t}{1-\beta_1^t}$ and $v_t = \frac{\beta_2 v_{t-1} + (1-\beta_2) g_t g_t^\top}{1-\beta_2^t}$
$\theta_{t+1} = \theta_t - \eta m_t / (v_t^{1/2} + \epsilon)$ (element-wise division)
end for
Output: final weight distribution $\mathcal{N}\left(\theta \,\middle|\, \left(\frac{N \mathrm{diag}(v_t^{1/2})}{N \mathrm{diag}(v_t^{1/2}) + 1/\sigma^2}\right) \theta_{t+1},\ \frac{1}{N \mathrm{diag}(v_t^{1/2}) + 1/\sigma^2}\right)$

Note that the denominators $(1 - \beta_1^t)$ and $(1 - \beta_2^t)$ in the updates for m_t and v_t, respectively, correct the initialization bias. These factors quickly converge to 1 and any effect on the final posterior variance quickly deteriorates. Thus, in practice, we can absorb those factors into the learning rate by using $\eta_t = \eta \sqrt{(1 - \beta_2^t)}/(1 - \beta_1^t)$ in place of η, as is usually done in many implementations of ADAM, including that in TensorFlow. Note, that a strict interpretation of N as the number of points, does not allow for learning in an online scenario. Also, note that with dropout the loss function landscape changes with each pass through the data. With this in mind a heuristic which we found experimentally successful is to set $N = t \times$ batch size.

5 Results

Having introduced our method for obtaining an approximate posterior over the weights of a NN. We now present experimental evidence to demonstrate the approximate posterior's effectiveness on MNIST classification and in a contextual bandit setting. The code to reproduce these experiments is available at `github.com/skezle/BADAM`. We present our method in comparison to popular methods

[2]For the same reason, EWMAs are popular in finance where one encounters noisy observations from non-stationary distributions.

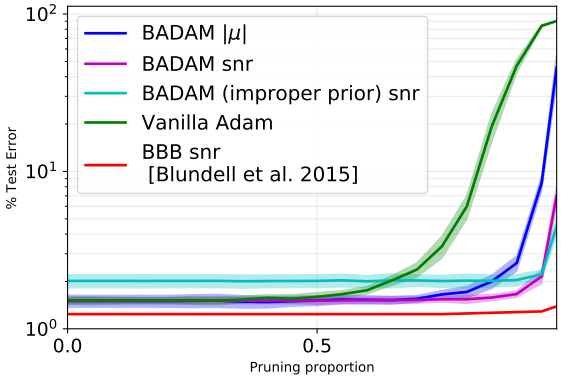


Figure 2: Weight pruning test accuracy of BADAM and BBB networks versus cut-off proportion. The curves are an average $\pm$ one standard error from 5 runs with random seeds.

such as BBB and MC DROPOUT. Our method has the advantage that the posterior distribution can be extracted for "free" from the standard ADAM algorithm.

We do not place much emphasis on the convergence properties of BADAM, as these properties are directly inherited from the underlying subgradient method which, in this case is ADAM. As the convergence of ADAM is well studied, and the algorithm is commonly used because of its good convergence properties. The primary goal of our experiments is to assess the quality of the uncertainties provided by BADAM.

5.1 Classification on MNIST

To assess the quality of the obtained uncertainties and to show that our posterior distributions are meaningful, we follow the weight-pruning experiment carried out in [11] on the MNIST classification problem. One cannot obtain uncertainties over the network weights with MC DROPOUT, as it implicitly integrates over them to obtain a predictive distribution, this is one advantage of BADAM and BBB. Given a posterior mean μ and standard deviation σ, we compute the signal-to-noise ratio as $|\mu|/\sigma$. To perform weight pruning of weights, we sort the weights by their signal-to-noise ratio and discard the fraction p of weights with the lowest values, by setting these weights to zero. As a baseline, we perform pruning on a model with constant variances, $\Sigma = \mathbb{1}$. Experimental details can be found in section B.

The bottom line of Figure 2 is that BADAM can produce high quality uncertainties seen as the pruning via signal-to-noise drop in the accuracy is smaller than pruning by the absolute value of the weight only. Additionally BADAM produces accurate predictions which are computationally cheap in comparison to BBB. We also found that by using BADAM with an improper prior and removing the initialization bias to the learning rate (using η and not η_t). Then we can achieve good uncertainties where the drop in performance due to high pruning rates is less marked. However, this is at a cost of achieving a sub-optimal local minimum in comparison to vanilla ADAM and our BADAM results with a Gaussian prior. BBB produces higher quality uncertainties as the drop off in the accuracy is small for higher pruning percentages than BADAM. However, BBB requires extensive hyperparameter tuning, long training times and is difficult to implement (we were unable to reproduce the experimental results from [11]). Also note the mean coefficient in equation 14 means that pruning BADAM with $|\mu|$ is more robust than pruning simply with the point estimates from ADAM.

5.2 Contextual Bandits

We demonstrate the effectiveness of the BADAM uncertainties by using the BNN in a contextual multi-armed bandit setting where and agent requires a good measure uncertainty for decision making. The contextual multi-armed bandit problem proceeds as follows, at times $t = 1, \ldots, T$ our agent will receive a context $X_t \in \mathbb{R}^d$ and will need to decide which action $a_t \in \mathcal{A}$ to pick to maximise a total reward $r = \sum_{t=1}^{T} r_t$. Our agent learns a function $f : (X_t, a_t) \to r_t \in \mathbb{R}$. Thompson sampling provides a Bayesian framework for our agent to manage the exploration exploitation dilema [29].

	Mushroom	Fiancial	Jester	Statlog	Adult	Covertype	Census
BADAM	0.77 ± 0.13	0.89 ± 0.03	0.46 ± 0.01	0.97 ± 0.02	0.19 ± 0.02	0.52 ± 0.06	0.44 ± 0.04
MC DROPOUT	0.79 ± 0.04	0.88 ± 0.03	0.46 ± 0.01	0.98 ± 0.02	0.18 ± 0.02	0.52 ± 0.06	0.43 ± 0.03
BBB	0.60 ± 0.09	0.09 ± 0.15	0.39 ± 0.10	0.69 ± 0.11	0.12 ± 0.01	0.43 ± 0.05	0.36 ± 0.08
Greedy	0.68 ± 0.13	0.91 ± 0.02	0.47 ± 0.01	0.97 ± 0.02	0.17 ± 0.02	0.57 ± 0.04	0.38 ± 0.03
Uniform	-0.93 ± 0.08	-0.00 ± 0.01	0.23 ± 0.00	0.14 ± 0.00	0.07 ± 0.00	0.14 ± 0.00	0.11 ± 0.00

Table 1: Contextual bandit cumulative rewards normalized by the optimal reward $\pm$ one standard error for different strategies. The results are an average and standard error from 10 runs. A detailed description of the datasets can be found in [32].

Thompson sampling requires that our model has some prior over the model parameters at t, $p(\theta_t)$. At each iteration our model samples from its prior and then greedily selects the action which maximizes the reward. The model then receives feedback for the selected decision and updates prior to obtain a posterior $p(\theta_{t+1}|X_{1:t}, a_{1:t})$. No feedback is observed for actions which are not selected. Thompson sampling has been shown to be an effective strategy in practice [30] and in theory [31]. The challenge in this setting is that at time t our agent's approximate prior $\tilde{p}(\theta_t)$ is used to obtain the reward point r_t, and subsequently used to update our model/calculate a posterior, crucially the data points r_j, $j \leq t$ are not i.i.d. In [32] it is shown that this feedback loop together with the use of an approximate posterior can lead to large disagreements between the model's posterior and the true posterior. This in turn can result in large cumulative regrets. Further, [32] empirically studies how different model's posterior approximations hold up on a variety of tasks and conclude that those algorithms which are unable to separate representation learning and posterior uncertainty construction perform poorly. Since the posterior uncertainty construction and representation learning in the BADAM algorithm are separate we postulate good performance in this contextual multi-armed bandit with Thompson sampling. Details of the experimental setup can be found in Section C.

The results in Table 1 show that the rewards from using BADAM are comparable to those obtained from using MC DROPOUT and the purely greedy bandit and outperforms the use of BBB in line with the results from [32]. Exploration is key in contextual bandits and reinforcement learning in general, and stops our agent becoming stuck exploiting suboptimal actions. Some real datasets do not require significant exploration, hence we use a synthetic data set which explicitly requires exploration to achieve optimality to test our algorithms. The wheel bandit is proposed and described in [32] and the amount of exploration is controlled via a parameter, δ. From small values of the exploration parameter BBB performs best and BADAM performs similarly to MC DROPOUT and the purely greedy bandit, for very high values of δ the exploration becomes very challenging and all algorithms struggle with the wheel bandit. See section C.2 for experimental details and Table 2 for the results.

6 Conclusion

In this paper, we introduce a novel approach to Bayesian learning for neural networks, derived from a new probabilistic interpretation of adaptive subgradient algorithms. In particular, we discuss how to refine this framework to ADAM and ADAGRAD. We focus on altering ADAM as the EWMA estimate of the curvature is preferable. Finally, we demonstrate empirically the performance of the posterior distribution on MNIST classification and on contextal bandit problems. BADAM is computationally efficient, like MC DROPOUT, but has the additional advantage of producing an explicit approximate posterior distribution like [11, 14], without the complexity of implementing and performing stochastic variational inference.

Finally, we want to emphasize that the key underlying principle of the Bayesian treatment we propose in this paper is to provide a measure of uncertainty over a neural network's weight parameters, and not just a better or faster (in convergence terms) point estimate thereof. The quality of this uncertainty metric can be assessed by pruning the weights: the more robust the classification error of a Bayesian algorithm for learning neural networks is to weight pruning, the better the quality of the uncertainty embedded in the corresponding (approximate) weight posterior distribution will be. An algorithm achieving a smaller error at higher pruning rates (relative to pruning by μ only)– even if its corresponding error rate at 0% pruning is less attractive – comes with genuinely desirable uncertainty estimates. This is clearly illustrated in Figure 2. The quality of the uncertainty can also be judged on their performance on a Thompson sampling problem where decisions are made according the

approximate posterior of our algorithms. That BADAM is able to achieve similar performance to standard benchmarks is testament to the quality of its approximate posterior distribution.

We hope that BADAM will find usage as a practical off-the-shelf Bayesian adaptive subgradient algorithm, providing posterior distributions with highly accurate confidence measures over neural-network parameters.

References

[1] David J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4:448–472, 1992.

[2] Geoffrey E. Hinton and Drew van Camp. Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. In *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, 1993.

[3] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

[4] José Miguel Hernández Lobato and Ryan P. Adams. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1861–1869, 2015.

[5] Daniel Soudry, Itay Hubara, and Ron Meir. Expectation Backpropagation: Parameter-Free Training of Multilayer Neural Networks with Continuous or Discrete Weights. In *Advances in Neural Information Processing Systems 27*, pages 963–971, 2014.

[6] José Miguel Hernández Lobato, Yingzhen Li, Mark Rowland, Thang Bui, Daniel Hernández-Lobato, and Richard Turner. Black-Box Alpha Divergence Minimization. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1511–1520, 2016.

[7] Anoop Korattikara Balan, Vivek Rathod, Kevin P. Murphy, and Max Welling. Bayesian Dark Knowledge. In *Advances in Neural Information Processing Systems 28*, 2015.

[8] Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[9] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[10] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming Catastrophic Forgetting by Incremental Moment Matching. In *Advances in Neural Information Processing Systems 30*, 2017.

[11] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[12] Alex Graves. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems 24*, 2011.

[13] Diederik P. Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. *CoRR*, abs/1506.02557, 2015.

[14] Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[15] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational Dropout Sparsifies Deep Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[16] Jan Achterhold, Jan M. Kohler, Anke Schmeink, and Tim Genewein. Variational Network Quantization. In *6th International Conference on Learning Representations*, 2018.

[17] Jiri Hron, Alexander G. de G. Matthews, and Zoubin Ghahramani. Variational Gaussian Dropout is not Bayesian. *arXiv:1711.02989 [stat.ML]*, 2017.

[18] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Bayesian Gradient Descent: Online Variational Bayes Learning with Increased Robustness to Catastrophic Forgetting and Weight Pruning. *arXiv:1803.10123 [stat.ML]*, 2018.

[19] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[20] Tijman Tieleman and Geoffrey Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[21] Diederik P. Kingma and Jimmy B. Lei. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[22] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic Gradient Descent as Approximate Bayesian Inference. *Journal of Machine Learning Research*, 18:1–35, 2017.

[23] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936, 2003.

[24] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. In *6th International Conference on Learning Representations*, 2018.

[25] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012.

[26] Timothy Dozat. Incorporating Nesterov Momentum into Adam. In *4th International Conference on Learning Representations – Workshop Track*, 2016.

[27] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *arXiv: 1612.00796v2*, 2017.

[28] Ferenc Huszár. On Quadratic Penalties in Elastic Weight Consolidation. 2017.

[29] William R Thompson. On The Likelihood That One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3-4):285–294, 12 1933.

[30] Olivier Chapelle and Lihong Li. An Empirical Evaluation of Thompson Sampling. 2011.

[31] Shipra Agrawal and Navin Goyal. Analysis of Thompson Sampling for the multi-armed bandit problem. *CoRR*, abs/1111.1797, 2011.

[32] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. *International Conference on Learning Representations, ICLR.*, 2018.

A Toy Regression Experiments

Data generation. Data is generated from the function $y = x + 0.3\sin(2\pi(x+\epsilon)) + 0.3\sin(4\pi(x+\epsilon)) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.02)$. Data for training is sampled from the range $(0.0, 0.5)$ while data used for evaluating the testing was sampled from the range $(-0.5, 1.2)$.

Model architecture. A four layer NN with ReLU activations and hidden state sixes of 100 neurons are used for our BADAM network and baseline MC DROPOUT and BBB networks. For training BADAM we use $\sigma_{prior} = 0.1$ a dropout rate of 0.05 and N (in equation 14) to the number of points in the training set multiplied by the number of epochs it trained for. Training involves gradient clipping such that the 2-norm of the gradients are not greater than 5. The MC DROPOUT network uses a dropout rate of 0.5. The BBB network uses a prior variance over the weights equal to $\sqrt{2/n_{in}}$ where n_{in} is the number of incoming nodes. We use an implementation based from [32]. The training set has 10000 samples and the MC DROPOUT network is trained for 100 epochs, while BBB and BADAM for 1000 epochs. The predictive distributions are generated from 10000 test points.

B MNIST Weight Pruning Experiments

Model setup. In order to make the results of our framework comparable to those obtained by the BBB algorithm, we replicate the experimental setup used in [11] except we preprocess the pixels by dividing values by 255 instead of by 126. The BADAM network uses dropout with a rate of 0.25. For BADAM we use a Gaussian prior over weights with mean 0 and prior variance of $\sigma_{prior} = 0.1$. We found that using a larger value of β_2 and removing the learning rate initialization bias produces good uncertainties (using η and not η_t). However, by not adapting the learning rate the BADAM optimization produces sub-optimal results when there is no pruning. To achieve good accuracies when there is no pruning and good uncertainty estimates for pruning we opt for a dual optimization approach. We run BADAM with default settings for 100 epochs to achieve a good solution. Then run BADAM again for 50 epochs to pick up curvature information around our local optimum employing $\beta_2 = 0.99996$ and removing the initialization bias to the learning rate. Our BBB implementation was unable to reproduce the high accuracies seen in the weight pruning experiments in [11], hence we report the results directly from the paper.

We also experimented with an improper prior and found that by removing the initialization bias to the learning rate and increasing $\beta_2 = 0.99999$ then one achieves good uncertainty estimates. However at the price of a less good local optimum than can otherwise be found with pure ADAM.

Experimental setup. Apart from the details provided in the previous subsection, the experimental configuration is identical to that used by the authors of [11]. That is, we use the same network architecture with 2 hidden layers made of 1200 units each with ReLU activation functions and softmax output layer. The total number of parameters is approximately 2.4 million. We use a training and test set of 60000 and 10000 images respectively.

C Contextual Bandits Experiments

C.1 Real datasets

Model step. We use the experimental setup described in an implementation provided by [32] for evaluation of our BADAM algorithm. A detailed description of the datasets used for our multi-armed bandit experiments can also be found in the appendix of [32]. We compare to BBB, MC DROPOUT an NN which greedily picks each action (named Greedy in Tables 1 and 2), and as a baseline an agent which uniformly samples actions. The neural networks architectures used for all networks are the same: 2 layers with 100 units each and ReLU activations, they regress contexts in $\mathbb{R}^d$ to outputs in $\mathbb{R}^{|\mathcal{A}|}$. For all non-variational methods the weights are initialized with $U[-0.3, 0.3]$, and use gradient clipping such that the 2-norm of the gradients are not greater than 5. In terms of hyperparameters, all networks use an initial learning rate of 0.1 with an inverse decaying schedule. MC DROPOUT and BBB uses an Adam optimizer and a dropout rate of 0.5 while the greedy network uses an RMSProp optimizer [20]. BADAM uses a Gaussian prior $\mathcal{N}(0, 0.1^2)$. BBB uses a Gaussian prior like that described in section A. We do not perform hyperparameter optimization similarly to [11].

Multi-armed bandits experimental details. The experimental setup for the multi-armed bandits proceeds as follows: at each round a new context from the dataset is presented to the bandit algorithm, we go through the dataset once. Each action is initially selected 3 times so that each agent has some initial information to learn from. Subsequently, actions are greedily chosen, and only the reward for the chosen action will be backpropagated upon training. In terms of training, all observed contexts,

	$\delta = 0.50$	$\delta = 0.70$	$\delta = 0.90$	$\delta = 0.95$	$\delta = 0.99$
BADAM	0.76 ± 0.02	0.74 ± 0.09	0.83 ± 0.07	0.34 ± 0.08	0.59 ± 0.01
MC_DROPOUT	0.76 ± 0.08	0.80 ± 0.08	0.66 ± 0.11	0.40 ± 0.12	0.61 ± 0.04
BBB	0.97 ± 0.02	0.95 ± 0.02	0.93 ± 0.02	0.41 ± 0.18	0.58 ± 0.00
Greedy	0.81 ± 0.17	0.85 ± 0.15	0.87 ± 0.09	0.38 ± 0.13	0.58 ± 0.00
Uniform	0.22 ± 0.00	0.23 ± 0.01	0.28 ± 0.01	0.35 ± 0.01	0.58 ± 0.01

Table 2: Contextual bandit cumulative rewards normalized by the optimal reward $\pm$ one standard error for different algorithms on the wheel bandit which has been run 10 times. The parameter δ controls the difficulty of exploration required by our agents, large δ the more difficult the exploration.

actions and rewards are stored in a buffer. The buffer is sampled to create batches used for training. Training occurred every $t_f = 20$ rounds for $t_s = 50$ minibatches using a batch size of 512 for all neural bandits.

The full datasets are used for all bandits experiments apart from the census and covertype datasets which are very large, we use a subset of $n = 10000$ points from these two datasets.

C.2 Wheel bandit

The wheel bandit is described in [32]. We use the same experimental setup for our BNN algorithms as described in the previous subsection except our BADAM prior is $\mathcal{N}(0, 0.2^2)$ to encourage a little more exploration in comparison to the real datasets. The context dimension is $\in \mathbb{R}^2$.