entropy

MDPI

Article

Gradient Regularization as Approximate Variational Inference

Ali Unlu [1] and Laurence Aitchison [2,*]

[1] Department of Infomatics, University of Sussex, Brighton BN1 9QJ, UK; a.unlu@sussex.ac.uk
[2] Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK
* Correspondence: laurence.aitchison@gmail.com

Abstract: We developed Variational Laplace for Bayesian neural networks (BNNs), which exploits a local approximation of the curvature of the likelihood to estimate the ELBO without the need for stochastic sampling of the neural-network weights. The Variational Laplace objective is simple to evaluate, as it is the log-likelihood plus weight-decay, plus a squared-gradient regularizer. Variational Laplace gave better test performance and expected calibration errors than maximum a posteriori inference and standard sampling-based variational inference, despite using the same variational approximate posterior. Finally, we emphasize the care needed in benchmarking standard VI, as there is a risk of stopping before the variance parameters have converged. We show that early-stopping can be avoided by increasing the learning rate for the variance parameters.

Keywords: variational inference; Laplace; Bayes; Bayesian neural networks

Citation: Unlu, A.; Aitchison, L. Gradient Regularization as Approximate Variational Inference. *Entropy* **2021**, *23*, 1629. https://doi.org/10.3390/e23121629

Academic Editors: Eric Nalisnick and Dustin Tran

Received: 20 September 2021
Accepted: 1 November 2021
Published: 3 December 2021

1. Introduction

Neural networks are increasingly being used in safety-critical settings, such as self-driving cars [1] and medical diagnosis [2]. In these settings, it is critical to be able to reason about uncertainty in the parameters of the network—for instance, so that the system is able to call for additional human input when necessary [3]. Several approaches to Bayesian inference in neural networks are available, including stochastic gradient Langevin dynamics [4] Laplace's method [5–7] and variational inference [8,9].

Here, we focus on combining the advantages of Laplace's method [5–7] and variational inference (VI; [10]). In particular, Laplace's method is very fast, as it begins by finding a mode using a standard gradient descent procedure, and then computes a local Gaussian approximate of the mode by performing a second-order Taylor expansion. However, as the mode is discovered by standard gradient descent, it may be a narrow mode that generalizes poorly [11]. In contrast, variational inference (VI; [8]) is slower, as it requires stochastic sampling of the weights, but that stochastic sampling forces it to find a broad, flat mode that presumably generalizes better.

We developed a new Variational Laplace (VL) method that combines the best of both worlds, giving a method that finds broad, flat modes even in the absence of the stochastic sampling. The resulting objective is composed of the log-likelihood, standard weight-decay regularization and a squared-gradient regularizer, which is weighted by the variance of the approximate posterior. VL displayed improved performance over VI and MAP in standard benchmark tasks.

Our squared gradient regularizer relates strongly to the effectiveness of stochastic gradient descent. In particular, recent work has shown that gradient descent implicitly uses a squared gradient regularizer [12], and that full-batch gradient descent with a squared gradient regularizer can recover much of the benefits of implicit regularization from minibatched stochastic gradient descent [13]. Our work implies that these regularizers can be interpreted as a form of approximate inference over the neural-network weights.

2. Background

2.1. Variational Inference (VI) for Bayesian Neural Networks

To perform variational inference for neural networks, we follow the usual approach [8,14], by using independent Gaussian priors, P and approximate posteriors Q for all parameters, $\mathbf{w}$:

$$\mathrm{P}(w_\lambda) = \mathcal{N}\left(w_\lambda; 0, s_\lambda^2\right) \tag{1}$$

$$\mathrm{Q}(w_\lambda) = \mathcal{N}\left(w_\lambda; \mu_\lambda, \sigma_\lambda^2\right) \qquad \text{equivalently} \qquad \mathrm{Q}(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2}$$

where μ_λ and σ_λ^2 are learned parameters of the approximate posterior, and where $\boldsymbol{\Sigma}$ is a diagonal matrix, with $\Sigma_{\lambda\lambda} = \sigma_\lambda^2$. We fit the approximate posterior by optimizing the evidence lower bound objective (ELBO) with respect to parameters of the variational posterior, μ_λ and σ_λ^2:

$$\mathcal{L}_{\mathrm{VI}} = \mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}\left[\log \mathrm{P}(\mathbf{y}|\mathbf{x}, \mathbf{w}) + \beta \sum_\lambda \log \frac{\log \mathrm{P}(w_\lambda)}{\log \mathrm{Q}(w_\lambda)}\right]. \tag{3}$$

Here, $\mathbf{x}$ is all training inputs, $\mathbf{y}$ is all training outputs, and β is the tempering parameter which is 1 for a close approximation to Bayesian inference, but is often set to smaller values to "temper" the posterior, which often improves empirical performance [15,16] and has theoretical justification as accounting for the data-curation process [17].

We need to optimize the expectation in Equation (3) with respect to the parameters of $\mathrm{Q}(\mathbf{w})$, the distribution over which the expectation is taken. To perform this optimization efficiently, the usual approach is to use the reparameterization trick [8,18,19]—we write $\mathbf{w}$ in terms of $\boldsymbol{\epsilon}$:

$$w_\lambda(\epsilon_\lambda) = \mu_\lambda + \sigma_\lambda \epsilon_\lambda \tag{4}$$

where $\epsilon_\lambda \sim \mathcal{N}(0, 1)$. Thus, the ELBO can be written as an expectation over $\boldsymbol{\epsilon}$:

$$\mathcal{L}_{\mathrm{VI}} = \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}\left[\log \mathrm{P}(\mathbf{y}|\mathbf{x}, \mathbf{w}(\boldsymbol{\epsilon})) + \beta \sum_\lambda \log \frac{\log \mathrm{P}(w_\lambda(\epsilon_\lambda))}{\log \mathrm{Q}(w_\lambda(\epsilon_\lambda))}\right]. \tag{5}$$

where the distribution over $\boldsymbol{\epsilon}$ is now fixed. Critically, now the expected gradient of the term inside the expectation is equal to the gradient of $\mathcal{L}_{\mathrm{VI}}$, so we can use samples of $\boldsymbol{\epsilon}$ to estimate the expectation.

2.2. Laplace's Method

Laplace's method [5–7] first finds a mode by doing gradient ascent on the log-joint:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} [\log \mathrm{P}(\mathbf{y}|\mathbf{x}, \mathbf{w}) + \log \mathrm{P}(\mathbf{w})] \tag{6}$$

and uses a Gaussian approximate posterior around that mode,

$$\mathrm{Q}(\mathbf{w}) = \mathcal{N}\left(\mathbf{w}; \mathbf{w}^*, -\mathbf{H}^{-1}(\mathbf{w}^*)\right) \tag{7}$$

where $\mathbf{H}(\mathbf{w}^*)$ is Hessian of the log-joint at $\mathbf{w}^*$.

3. Related Work

There is past work on Variational Laplace [20–22], which learns the mean parameters, $\boldsymbol{\mu}$, of a Gaussian approximate posterior,

$$\mathrm{Q}_{\boldsymbol{\mu}}(\mathbf{w}) = \mathcal{N}\left(\mathbf{w}; \boldsymbol{\mu}, -\mathbf{H}^{-1}(\boldsymbol{\mu})\right) \tag{8}$$

and obtains the covariance matrix as a function of the mean parameters using the Hessian, as in Laplace's method. However, instead of taking the approximation to be centered around a MAP solution, $\mathbf{w}^*$, they take the approximate posterior to be centered on learned mean parameters, $\boldsymbol{\mu}$. Importantly, they simplify the ELBO by substituting this approximate posterior into Equation (3), and approximating the log-joint using its Taylor series expansion. Ultimately they obtain

$$\mathcal{L}_{\text{VI}} \approx \log \mathrm{P}(\mathbf{y}|\mathbf{w}{=}\boldsymbol{\mu}, \mathbf{x}) + \log \mathrm{P}(\mathbf{w}{=}\boldsymbol{\mu}) - \tfrac{1}{2} \log|\mathbf{H}(\boldsymbol{\mu})| + \text{const.} \tag{9}$$

However, there are two problems with this approach when applied to neural networks. First, the algebraic manipulations required to derive Equation (9) require the full $N \times N$ Hessian, $\mathbf{H}(\boldsymbol{\mu})$, for all N parameters, and neural networks have too many parameters for this to be feasible. Second, the $\log|\mathbf{H}(\boldsymbol{\mu})|$ term in Equation (9) cannot be minibatched, as we need the full sum over minibatches inside the log to compute the Hessian:

$$\log|\mathbf{H}(\boldsymbol{\mu})| = \log\left|\textstyle\sum_j \mathbf{H}_j(\boldsymbol{\mu})\right|, \tag{10}$$

where $\mathbf{H}_j(\boldsymbol{\mu})$ is the contribution to the Hessian from an individual minibatch. Due to these issues, past Variational Laplace methods did not scale to large neural networks.

An alternative deterministic approach to variational inference in Bayesian neural networks, approximates the distribution over activities induced by stochasticity in the weights [23]. Unfortunately, it is important to capture the covariance over features induced by stochasticity in the weights. In fully connected networks, this is feasible, as we usually have a small number of features at each layer. However, in convolutional networks, we have a large number of features, `channels` $\times$ `height` $\times$ `width`. In the lower layers of a ResNet, we may have 64 channels and a 32×32 feature map, resulting in $64 \times 32^2 = 65{,}536$ features and a $65{,}536 \times 65{,}536$ covariance matrix. These scalability issues prevented them from applying their approach to convolutional networks. In contrast, our approach is highly scalable and readily applicable to the convolutional setting. Subsequent work such as Haußmann et al. [24] introduced other deterministic approximations, based on decomposing the relu into a linear function and a Heaviside step. However, their approach had errors of $\sim$30% on CIFAR-10.

Ritter et al. [7] and MacKay [25] used Laplace's method in Bayesian neural networks, by first finding the mode by doing gradient ascent on the log-joint probability, and expanding around that mode. As usual for Laplace's method, they risk finding a narrow mode that generalizes poorly. In contrast, we find a mode using an approximation to the ELBO that takes the curvature into account and hence is biased towards broad, flat modes that presumably generalise better.

Our approach gives a squared-gradient regularizer that is similar to those discovered in past work [12,26]. They showed that squared-gradient regularizers connect to gradient descent, in that approximation errors due to finite-step sizes in gradient-descent imply an effective squared gradient regularization. The similarity of our objectives raises profound questions about the extent to which gradient descent can be said to perform Bayesian inference. That said there are three key differences. First, their approach connects full-batch gradient descent to squared-gradient regularizers. Of course, most neural network training is stochastic gradient descent based on minibatches. Given that the stationary distribution of SGD is isotropic Gaussian (in a quadratic loss-function; Appendix A), we are able to connect *stochastic* gradient descent to squared gradient regularizers, and hence to approximate variational inference. This is especially important in light of recent work showing full-batch gradient descent gives poor performance, but that performance can be improved by including explicit squared gradient regularization. Our work indicates that explicit squared gradient regularization is mimicking the implicit regularization from stochastic gradient descent. First, our method uses the Fisher, (i.e., the gradients for data sampled from the model) whereas their approach uses the empirical Fisher, (i.e., gradients for the observed data) to form the squared gradient regularizer [27]. Second, our approach

gives a principled method to learn a separate weighting for the squared-gradient for each parameter, whereas the connection to SGD forces Barrett and Dherin [12] to use a uniform weighting across all parameters.

Our work differs from, e.g., Khan et al. [28] by explicitly providing a simply implemented loss-function in terms of a squared-gradient regularizer, instead of working with NTK-inspired approximations to the Hessian.

Other approaches include "Broad Bayesian Learning" [29], which optimizes the architecture of a Bayesian neural network, exploiting information from previously trained but different networks. Of course, quantification of uncertainty for Bayesian neural networks is always fraught [30]. As such, we followed standard practice in the literature of reporting OOD detection performance and a measure of calibration accuracy [31].

4. Methods

To combine the best of VI and Laplace's method, we begin by noting that the ELBO can be rewritten in terms of the KL divergence between the prior and approximate posterior:

$$\mathcal{L}_{\mathrm{VI}} = \mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}[\log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w})] - \beta \sum_{\lambda} \mathrm{D}_{\mathrm{KL}}(\mathrm{Q}(w_\lambda)||\,\mathrm{P}(w_\lambda)), \tag{11}$$

where the KL-divergence can be evaluated analytically:

$$D_{\mathrm{KL}}(\mathrm{Q}(w_\lambda)||\,\mathrm{P}(w_\lambda)) = \frac{1}{2}\left(\frac{\sigma_\lambda^2 + \mu_\lambda^2}{s_\lambda^2} - 1 + \log\frac{s_\lambda^2}{\sigma_\lambda^2}\right). \tag{12}$$

As such, the only term we need to approximate is the expected log-likelihood.

To approximate the expectation, we begin by taking a second-order Taylor series expansion of the log-likelihood around the current settings of the mean parameters, $\boldsymbol{\mu}$:

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}\left[\log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w})\right] \approx \log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w}{=}\boldsymbol{\mu}) + \mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}\left[\textstyle\sum_{j=1}^{B} \mathbf{g}_j^T(\mathbf{w}-\boldsymbol{\mu})\right] \\ + \mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}\left[\tfrac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^T\mathbf{H}(\mathbf{w}-\boldsymbol{\mu})\right]\end{aligned} \tag{13}$$

where B is the number of minibatches, $\mathbf{g}_j$ is the gradient for minibatch j and $\mathbf{H}$ is the Hessian for the full dataset:

$$g_{j;\lambda} = \frac{\partial}{\partial w_\lambda}\left[\log \mathrm{P}\left(\mathbf{y}_j|\mathbf{x}_j,\mathbf{w}\right)\right] \tag{14}$$

$$H_{\lambda,\nu} = \frac{\log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w})}{\partial w_\lambda \partial w_\nu}. \tag{15}$$

Here, $\mathbf{x}$ and $\mathbf{y}$ are the the inputs and outputs for the full dataset, whereas $\mathbf{x}_j$ and $\mathbf{y}_j$ are the inputs and outputs for minibatch j. Now we consider the expectation of each of these terms under the approximate posterior, $\mathrm{Q}(\mathbf{w})$. The first term is constant and independent of $\mathbf{w}$. The second (linear) term is zero, because the expectation of $(\mathbf{w}-\boldsymbol{\mu})$ under the approximate posterior is zero

$$\mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}\left[\mathbf{g}_j^T(\mathbf{w}-\boldsymbol{\mu})\right] = \mathbf{g}_j^T \mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}[(\mathbf{w}-\boldsymbol{\mu})] = 0. \tag{16}$$

The third (quadratic) term might at first appear difficult to evaluate because it involves $\mathbf{H}$, the $N \times N$ matrix of second derivatives, where N is the number of parameters in the model. However, using properties of the trace, and noting that the expectation of $(\mathbf{w}-\boldsymbol{\mu})(\mathbf{w}-\boldsymbol{\mu})^T$ is the covariance of the approximate posterior, we obtain

$$\mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}\left[\tfrac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^T\mathbf{H}(\mathbf{w}-\boldsymbol{\mu})\right] = \mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}\left[\tfrac{1}{2}\,\mathrm{Tr}\Big(\mathbf{H}(\mathbf{w}-\boldsymbol{\mu})(\mathbf{w}-\boldsymbol{\mu})^T\Big)\right] = \tfrac{1}{2}\,\mathrm{Tr}(\mathbf{H}\boldsymbol{\Sigma}) \tag{17}$$

writing the trace in index notation, and substituting for the (diagonal) posterior covariance, $\mathbf{\Sigma}$:

$$\tfrac{1}{2}\operatorname{Tr}(\mathbf{H\Sigma}) = \tfrac{1}{2}\sum_{\lambda\nu} H_{\lambda\nu}\Sigma_{\lambda\nu} = \tfrac{1}{2}\sum_{\lambda} H_{\lambda\lambda}\sigma_\lambda^2. \tag{18}$$

Thus, our first approximation of the expected log-likelihood is

$$\mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}[\log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w})] \approx \log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w}{=}\boldsymbol{\mu}) + \tfrac{1}{2}\textstyle\sum_\lambda \sigma_\lambda^2 H_{\lambda\lambda}, \tag{19}$$

and substituting this into Equation (11) gives

$$\mathcal{L}_{\mathrm{VI}} \;\approx\; \mathcal{L}_{\mathrm{VL(H)}} \;=\; \log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w}{=}\boldsymbol{\mu}) + \tfrac{1}{2}\textstyle\sum_\lambda \sigma_\lambda^2 H_{\lambda\lambda} - \beta \sum_\lambda \mathrm{D}_{\mathrm{KL}}(\mathrm{Q}(w_\lambda)||\,\mathrm{P}(w_\lambda)). \tag{20}$$

This resolves most of the issues with the original Variational Laplace method: it requires only the diagonal of the Hessian, it can be minibatched and it does not blow up if $H_{\lambda\lambda}$ is zero.

4.1. Pathological Optima When Using the Hessian

However, a new issue arises: $H_{\lambda\lambda}$ is usually negative, in which case the approximation in Equation (20) can be expected to work well. However there is nothing to stop $H_{\lambda\lambda}$ from becoming positive. Usually if we, e.g., took the log-determinant of the negative Hessian, this would immediately break the optimization process (as we would be taking the logarithm of a negative number). However, in our context, there is no immediate issue as Equation (20) takes on a well-defined value even when one or more $H_{\lambda\lambda}$'s are positive. That said, we rapidly encounter similar issues as we get pathological optimal values of σ_λ^2. In particular, picking out the terms in the objective that depend on σ_λ^2, absorbing the other terms into the constant, and taking $\beta = 1$ for simplicity, we have

$$\mathcal{L}_{\mathrm{VL(H)}} = \tfrac{1}{2}\textstyle\sum_\lambda \left(-\left(\frac{1}{s_\lambda^2} - H_{\lambda\lambda} \right)\sigma_\lambda^2 + \log \sigma_\lambda^2 \right) + \text{const.} \tag{21}$$

Thus, the gradient wrt a single variance parameter is

$$\frac{\partial}{\partial \sigma_\lambda^2}\mathcal{L}_{\mathrm{VL(H)}} = \tfrac{1}{2}\left(-\left(\frac{1}{s_\lambda^2} - H_{\lambda\lambda} \right) + \frac{1}{\sigma_\lambda^2} \right). \tag{22}$$

In the typical case, $H_{\lambda\lambda}$ is negative so $\left(\frac{1}{s_\lambda^2} - H_{\lambda\lambda} \right)$ is positive, and we can find the optimum by solving for the value of σ_λ^2 where the gradient is zero:

$$\sigma_\lambda^2 = \frac{1}{\frac{1}{s_\lambda^2} - H_{\lambda\lambda}}. \tag{23}$$

However, if $H_{\lambda\lambda}$ is positive and sufficiently large, $H_{\lambda\lambda} > \frac{1}{s_\lambda^2}$, then $\left(\frac{1}{s_\lambda^2} - H_{\lambda\lambda} \right)$ becomes negative, and not only is the mode in Equation (23) undefined, but the gradient is always positive:

$$0 < \frac{\partial}{\partial \sigma_\lambda^2}\mathcal{L}_{\mathrm{VL(H)}} = \tfrac{1}{2}\left(-\left(\frac{1}{s_\lambda^2} - H_{\lambda\lambda} \right) + \frac{1}{\sigma_\lambda^2} \right). \tag{24}$$

as both terms in the sum: $-\left(\frac{1}{s_\lambda^2} - H_{\lambda\lambda} \right)$ and $\frac{1}{\sigma_\lambda^2}$ are positive. As such, when $H_{\lambda\lambda} > \frac{1}{s_\lambda^2}$, the variance, σ_λ^2 grows without bound.

4.2. Avoiding Pathologies with the Fisher

To avoid pathologies arising from the fact that the Hessian is not necessarily negative definite, a common approach is to approximate the Hessian using the Fisher information matrix:

$$-\mathbf{H} \approx \mathbf{F} = \sum_{j=1}^{B} \mathop{\mathbb{E}}_{\mathrm{P}(\tilde{\mathbf{y}}_j|\mathbf{x}_j,\mathbf{w}=\boldsymbol{\mu})} \left[\tilde{\mathbf{g}}_j(\tilde{\mathbf{y}}_j)\tilde{\mathbf{g}}_j^T(\tilde{\mathbf{y}}_j)\right]. \tag{25}$$

Importantly, $\tilde{\mathbf{g}}$ is the gradient of the log-likelihood for data sampled from the model, $\tilde{\mathbf{y}}_j$, *not* for the true data:

$$\tilde{g}_{j;\lambda}(\tilde{\mathbf{y}}_j) = \frac{\partial}{\partial w_\lambda}\left[\log \mathrm{P}\left(\tilde{\mathbf{y}}_j|\mathbf{x}_j,\mathbf{w}\right)\right]. \tag{26}$$

This gives us the Fisher, which is a commonly used and well-understood approximation to the Hessian [27]. Importantly, this contrasts with the empirical Fisher [27], which uses the gradient conditioned on the actual data (and not data sampled from the model):

$$\mathbf{F}_{\mathrm{emp}} = \sum_{j=1}^{B} \mathbf{g}_j \mathbf{g}_j^T, \tag{27}$$

which is problematic, because there is a large rank-1 component in the direction of the mean gradient, which disrupts the estimated matrix specifically in the direction of interest for problems such as optimization [27].

Using the Fisher information (Equation (25)) in Equation (19), we obtain an approximate expected log-likelihood:

$$\mathop{\mathbb{E}}_{\mathrm{Q}(\mathbf{w})}\left[\log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w})\right] \approx \log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w}{=}\boldsymbol{\mu}) - \tfrac{1}{2}\textstyle\sum_\lambda \sigma_\lambda^2 \sum_{j=1}^{B} \tilde{g}_{j;\lambda}^2. \tag{28}$$

Substituting this into Equation (11) gives us the final VL objective, $\mathcal{L}_{\mathrm{VL}}$, which is an approximation of the ELBO:

$$\mathcal{L}_{\mathrm{VI}} \approx \mathcal{L}_{\mathrm{VL}} = \log \mathrm{P}(\mathbf{y}|\mathbf{x},\mathbf{w}{=}\boldsymbol{\mu}) - \tfrac{1}{2}\textstyle\sum_\lambda \sigma_\lambda^2 \sum_{j=1}^{B} \tilde{g}_{j;\lambda}^2 - \beta \sum_\lambda \mathrm{D}_{\mathrm{KL}}(\mathrm{Q}(w_\lambda)||\,\mathrm{P}(w_\lambda)). \tag{29}$$

In practice, we typically take the objective for a minibatch, divided by the number of datapoints in a minibatch, S:

$$\tfrac{1}{S}\mathcal{L}_{\mathrm{VL};j} = \tfrac{1}{S}\log \mathrm{P}\left(\mathbf{y}_j|\mathbf{x}_j,\mathbf{w}{=}\boldsymbol{\mu}\right) - \tfrac{S}{2}\textstyle\sum_\lambda \sigma_\lambda^2 \left(\frac{1}{S}\tilde{g}_{j;\lambda}\right)^2 - \frac{\beta}{2SB}\sum_\lambda \left(\frac{\sigma_\lambda^2+\mu_\lambda^2}{s_\lambda^2} - 1 + \log\frac{s_\lambda^2}{\sigma_\lambda^2}\right), \tag{30}$$

where $\left(\frac{1}{S}\tilde{g}_{j;\lambda}\right)$ are the gradients of the log-likelihood for the minibatch averaged across datapoints, i.e., the gradient of $\frac{1}{S}\log \mathrm{P}\left(\tilde{\mathbf{y}}_j|\mathbf{x}_j,\mathbf{w}{=}\boldsymbol{\mu}\right)$. Remember B is the number of minibatches so SB is the total number of training datapoints.

4.3. Constraints on the Network Architecture

Importantly, here the regularizer is the squared gradient of the loss with respect to the parameters. As such, computing the loss implicitly involves a second-derivative of the log-likelihood, and we therefore cannot use piecewise linear activation functions such as ReLU, which have pathological second derivatives. In particular, the second derivative has a delta-function "spike" at zero:

$$\frac{d^2}{dx}\phi(x) = \frac{d}{dx}\left[\frac{d}{dx}\phi(x)\right] = \frac{d}{dx}\Theta(x) = \delta(x) \tag{31}$$

where ϕ is the relu nonlinearity, $\Theta(x)$ is the Heaviside step function which is zero for $x < 0$ and one for $0 < x$, and $\delta(x)$ is the Dirac delta function. As the function is almost never evaluated at exactly zero, it is not possible to sensibly take into account the contribution of the infinitely high spike in the second derivative at zero. Interestingly, this issue is very similar to the one that turns up when differentiating step (i.e., $\Theta(x)$) activations—the derivative is well-defined and zero almost everywhere. The issue is that there are delta-function spikes in the gradient at zero that gradient descent cannot reasonably work with. Instead, we used a softplus activation function, but any activation with well-behaved second derivatives is admissible.

5. Results

We compared MAP, VI and our method (VL) on four different datasets (CIFAR-10, CIFAR-100 [32], SVHN [33] and fashion-MNIST [34] MIT Licensed) using a PreactResNets-18 [35] with an initial learning rate of 1E-4, which decreased by a factor of 10 after 100 and 150 epochs and a batch size of 128 with all the other optimizer hyperparameters set to their default values. We tried two variants of variational inference: evaluating test-performance using the mean network, VI (mean), and evaluating test performance by drawing 10 samples from the approximate posterior, VI (sampled). We swept across different degrees of posterior tempering, β. Using $\beta < 1$ is normatively justified in the Bayesian framework as accounting for the effect of data curation [17]. For many values of β VL gave better test accuracies, test log-likelihoods and expected calibration errors [31,36] than VI or MAP inference (Figure 1). Importantly though, for the optimal value of β, VL almost always gave better performance on these metrics (Table 1). These experiments took ∼480 GPU hours, and were run on a mixture of nVidia 1080 and 2080 GPUs in an internal cluster.

Table 1. Best values test NLL, test accuracy and ECE for a variety of datasets, as we used different values of the tempering parameter, β.

Dataset	Method	Test NLL	Test Acc.	ECE
CIFAR-10	VL	**0.23**	**92.4%**	**0.017**
	VI (Mean)	0.37	91.1%	0.053
	VI (10 Samples)	0.35	90.2%	0.044
	MAP	0.43	90.8%	0.058
CIFAR-100	VL	**1.00**	**71.4%**	**0.024**
	VI (Mean)	1.29	68.8%	0.100
	VI (10 Samples)	1.49	67.3%	0.026
	MAP	1.61	67.5%	0.159
SVHN	VL	**0.14**	**97.1%**	**0.009**
	VI (Mean)	0.16	96.3%	0.012
	VI (10 Samples)	0.22	95.5%	0.022
	MAP	0.24	95.7%	0.028
Fashion MNIST	VL	**0.16**	**94.6%**	**0.010**
	VI (Mean)	0.23	94.0%	0.034
	VI (10 Samples)	0.29	93.6%	0.016
	MAP	0.29	93.6%	0.096

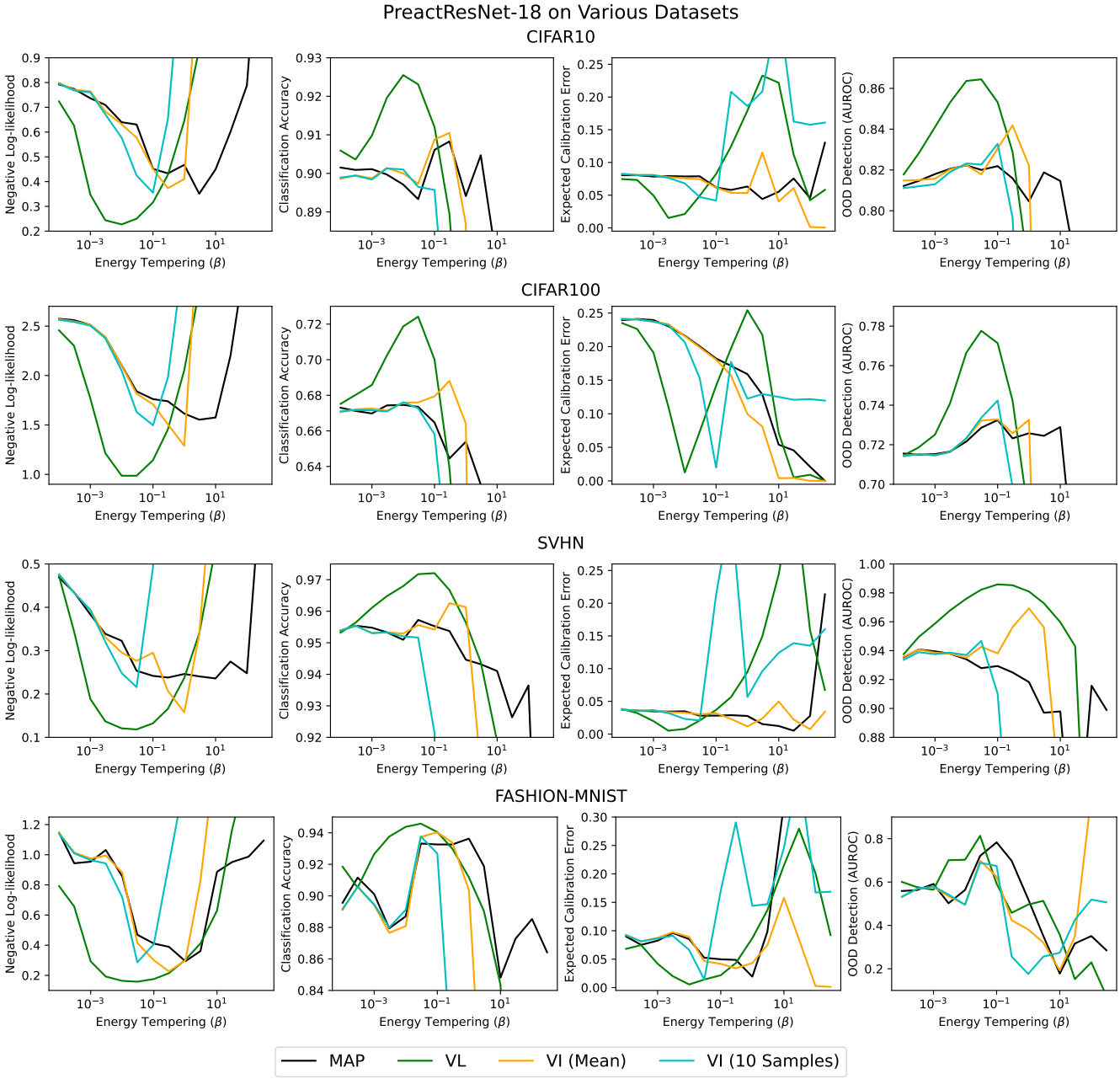

Figure 1. Training a PreactResNet-18 on various datasets, displaying the test accuracy, test log-likelihood, expected calibration error (ECE) [31,36] and OOD detection metric (AUROC) for CIFAR-10, CIFAR-100, SVHN and fashion MNIST. Downsampled Imagenet [37] was used as OOD data. See Appendix B.

The runtimes of the methods are listed in Table 2. VL or gradient regularization was around a factor of three slower than either VI or MAP due to the need to compute second-derivatives. It is still eminently feasible, especially in comparison to past methods for deterministic variational inference that have fundamental difficulties in scaling to convolutional networks [23]. Furthermore, we did not find that increasing the number of epochs improved performance either for VI or MAP, as we were already training for convergence.

Table 2. Time per epoch for different methods on CIFAR-10.

Method	Time per Epoch (s)
VL	114.9
VI	43.2
MAP	41.8

Early-Stopping and Poor Performance in VI

Before performing comparisons where we learn the approximate posterior variance, it is important to understand the pitfalls when optimizing variational Bayesian neural networks using adaptive optimizers such as Adam. In particular, there is a strong danger of stopping the optimization before the variances have converged. To illustrate this risk, note that Adam [38] updates take the form

$$\Delta\theta = \eta \frac{m}{\sqrt{v} + \epsilon} \tag{32}$$

where η is the learning rate, m is an unbiased estimator of the mean gradient, $\langle g \rangle$, v is an unbiased estimator of the squared gradient, $\langle g^2 \rangle$ and ϵ is a small positive constant to avoid divide-by-zero. The magnitude of the updates, $|\Delta\theta|$, is maximized by having exactly the same gradient on each step, in which case, neglecting ϵ, we have $|\Delta\theta| = \eta$. As such, with a learning rate of $\eta = 10^{-4}$, a training set of 50,000 and a batch size of 128 parameters can move at most $50{,}000/128 \times 10^{-4} \approx 0.04$ per epoch. Doing 100 epochs at this learning rate, a parameter can change by at most 4 over the 100 epochs before the first learning rate step.

This is fine for the weights, which typically have very small values. However, the underlying parameters used for the variances typically take on larger values. In our case, we will use $\log \sigma_\lambda$ as the parameter, and initialize it to three less than the prior standard deviation, $\log s_\lambda - 3$. To ensure reasonable convergence, $\log \sigma_\lambda$ should be able to revert back to the prior, implying that it must be able to change by at least three during the course of training. Unfortunately, 3 is very close to the maximum possible change of 4, raising the possibility that the variance parameters will not actually converge. To check whether early-stopping was indeed an issue, we plotted the (tempered) ELBO for VI (Figure 2A) and VL (Figure 2B). For VI (Figure 2A) with the standard setup (lightest line with a learning rate multiplier of 1), the ELBO clearly has not converged at 100 epochs, indicating early-stopping. Notably, this was still an issue with VL (Figure 2B), especially if we were to train for fewer epochs. However, the effect was smaller for VL, which may have been because the gradients were more consistent, as it did not sample the weights. These issues can be rectified by increasing the learning rate specifically for the $\log \sigma_\lambda$ parameters (darker lines).

We then plotted the test log-likelihood (Figure 2C), test accuracy (Figure 2D) and ELBO (Figure 2E) against the learning rate multiplier. Again, the performance for VL (orange) was reasonably robust to changes in the learning rate multiplier. However, the performance of VI (blue) was very sensitive to the multiplier: as the multiplier increased, test performance fell but the ELBO rose. As we ultimately care about test performance, these results would suggest that we should use the lowest multiplier (1), and accept the possibility of early-stopping. That may be a perfectly good choice in many cases. However, VI is supposed to be an approximate Bayesian method, and using an alternative form for the ELBO,

$$\mathcal{L}_{\mathrm{VI}} = \log \mathrm{P}(\mathbf{y}|\mathbf{x}) - D_{\mathrm{KL}}(\mathrm{Q}(\mathbf{w}) || \mathrm{P}(\mathbf{w}|\mathbf{y}, \mathbf{x})), \tag{33}$$

we can see that the ELBO measures KL-divergence between the true and approximate posterior, and hence the quality of our approximate Bayesian inference. As such, very poor ELBOs imply that the KL-divergence between the true and approximate posterior is very large, and hence the "approximate posterior" is no longer actually approximating the true posterior. As such, if we are to retain a Bayesian interpretation of VI, we need

to use larger learning rate multipliers which give better values for the ELBO (Figure 2E). However, in doing that, we get worse test performance (Figure 2C,D). This conflict between approximate posterior quality and test performance is very problematic: the Bayesian framework would suggest that as Bayesian inference becomes more accurate, performance should improve, whereas for VI, performance gets worse. Concretely, by initializing $\log \sigma_\lambda$ to a small value and then early-stopping, we leave $\log \sigma_\lambda$ at a small value through training, in which case VI becomes equivalent to MAP inference with a negligibly small amount of noise added to the weights. We would therefore expect early-stopped VI to behave (and be) very similar to MAP inference.

In subsequent experiments, we chose to use a learning rate multiplier of 10, as this largely eliminated early-stopping (though see VI with $\beta = 0.1$; Figure 2E). Overall, this indicates that we have to be very careful to avoid early stopping when running standard, sampling-based variational inference.

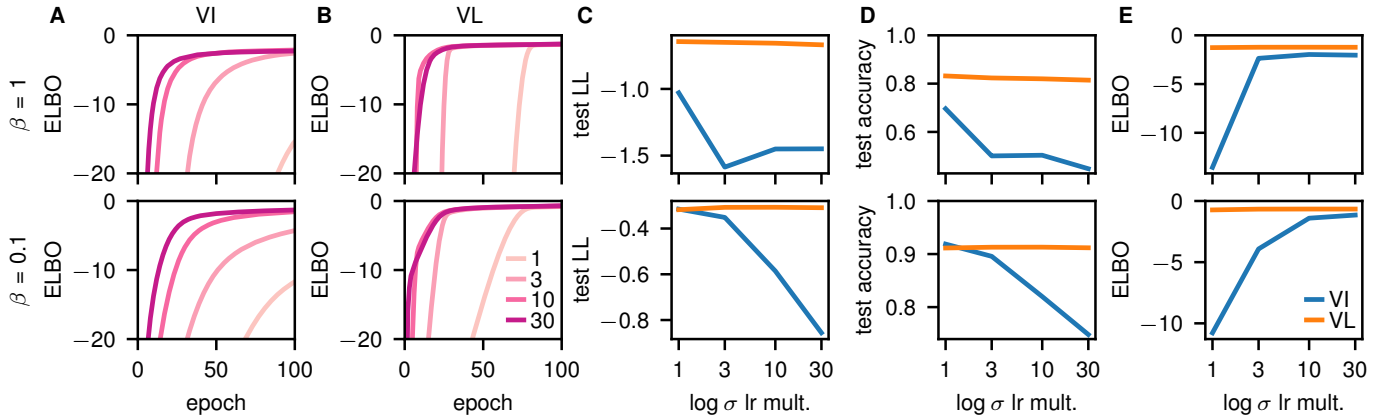

Figure 2. Analysis of early stopping in VI and VL. The first row is untempered ($\beta = 1$), and the second row is tempered ($\beta = 0.1$). (**A**) ELBO over epochs 0–100 (with the highest initial learning rate) for VI. Different lines correspond to networks with learning rate multipliers for $\log \sigma_\lambda$ of 1, 3, 10 and 30. (**B**) As (**A**), but for VL. **CDE** Final test-log-likelihood (**C**), test accuracy (**D**) and ELBO (**E**) after 200 epochs for different learning rate multipliers.

6. Conclusions

We gave a novel Variational Laplace approach to inference in Bayesian neural networks which combines the best of previous approaches based on variational inference and Laplace's Method. This method gave excellent empirical performance compared to VI.

Author Contributions: Conceptualization, L.A.; methodology, A.U.; software, A.U.; validation, L.A.; formal analysis, L.A.; investigation, A.U.; resources, L.A.; data curation, A.U.; writing—original draft preparation, L.A.; writing—review and editing, L.A.; visualization, A.U.; supervision, L.A.; project administration, L.A.; funding acquisition, L.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Code is available at https://github.com/LaurenceA/fitr (accessed on 2 December 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. The Stationary Distribution of SGD

We sought to relate these gradient regularizers back to work on SGD. In particular, we looked to work on the stationary distribution of SGD, which noted that under quadratic losses functions, SGD samples an *isotropic* Gaussian (i.e., with covariance proportional to the identity matrix). In particular, consider a loss function which is locally closely approximated by a quadratic. Without loss of generality, we consider a mode at $\mathbf{w} = \mathbf{0}$,

$$\log \mathrm{P}\left(\{y_i\}_{i=1}^P | \{x_i\}_{i=1}^P, \mathbf{w}\right) = -\tfrac{P}{2}\mathbf{w}^T\mathbf{H}\mathbf{w} + \text{const}. \tag{A1}$$

where P is the total number of datapoints. Typically, the objective used in SGD is the loss for a minibatch of size S_{SGD}. Following Mandt et al. [39], we use the Fisher information to identify the noise in the minibatch gradient,

$$\frac{\partial}{\partial \mathbf{w}}\left[\tfrac{1}{S_{\text{SGD}}} \log \mathrm{P}\left(\mathbf{y}_j | x_j, \mathbf{w}\right)\right] = -\mathbf{H}\mathbf{w} + \tfrac{1}{\sqrt{S_{\text{SGD}}}}\mathbf{H}^{1/2}\boldsymbol{\xi}(t), \tag{A2}$$

where $\boldsymbol{\xi}(t)$ is sampled from a standard IID Gaussian. For SGD, this gradient is multiplied by a learning rate, η_{SGD},

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_{\text{SGD}}\mathbf{H}\mathbf{w}(t) + \tfrac{\eta_{\text{SGD}}}{\sqrt{S_{\text{SGD}}}}\mathbf{H}^{1/2}\boldsymbol{\xi}(t), \tag{A3}$$

This is an multivariate Gaussian autoregressive process, so we can solve for the stationary distribution of the weights. In particular, we note that the covariance at time $t+1$ is

$$\begin{aligned}
\mathbb{C}[\mathbf{w}(t+1)] &= \mathbb{E}\left[\left(\mathbf{w}(t) - \eta_{\text{SGD}}\mathbf{H}\mathbf{w}(t) - \tfrac{\eta_{\text{SGD}}}{\sqrt{S_{\text{SGD}}}}\mathbf{H}^{1/2}\boldsymbol{\xi}(t)\right)^T\left(\mathbf{w}(t) - \eta_{\text{SGD}}\mathbf{H}\mathbf{w}(t) - \tfrac{\eta_{\text{SGD}}}{\sqrt{S_{\text{SGD}}}}\mathbf{H}^{1/2}\boldsymbol{\xi}(t)\right)\right] \\
\mathbb{C}[\mathbf{w}(t+1)] &= (\mathbf{I} - \eta_{\text{SGD}}\mathbf{H})^T\,\mathbb{C}[\mathbf{w}(t)](\mathbf{I} - \eta_{\text{SGD}}\mathbf{H}) + \tfrac{\eta_{\text{SGD}}^2}{S_{\text{SGD}}}\mathbf{H}
\end{aligned} \tag{A4}$$

Following Mandt et al. [39], when the learning rate is small, the quadratic term can be neglected.

$$\mathbb{C}[\mathbf{w}(t+1)] \approx \mathbb{C}[\mathbf{w}(t)] - \eta_{\text{SGD}}\mathbf{H}\,\mathbb{C}[\mathbf{w}(t)] - \eta_{\text{SGD}}\,\mathbb{C}[\mathbf{w}(t)]\mathbf{H} + \tfrac{\eta_{\text{SGD}}^2}{S_{\text{SGD}}}\mathbf{H} \tag{A5}$$

We then solve for steady-state in which $\boldsymbol{\Sigma} = \mathbb{C}[\mathbf{w}(t+1)] = \mathbb{C}[\mathbf{w}(t)]$,

$$0 \approx -\eta_{\text{SGD}}\left(\mathbf{H}^T\boldsymbol{\Sigma}_{\text{SGD}} + \boldsymbol{\Sigma}_{\text{SGD}}\mathbf{H}\right) + \tfrac{\eta_{\text{SGD}}^2}{S_{\text{SGD}}}\mathbf{H} \tag{A6}$$

so,

$$\boldsymbol{\Sigma} \approx \tfrac{\eta_{\text{SGD}}}{2S_{\text{SGD}}}\mathbf{I}. \tag{A7}$$

Appendix B. Varying the Batch Size

Here, we vary the batch size. We used a batch size of 128 in the main text. We found that batch sizes of 64 and 256 have no effect on the relative performance of the methods.

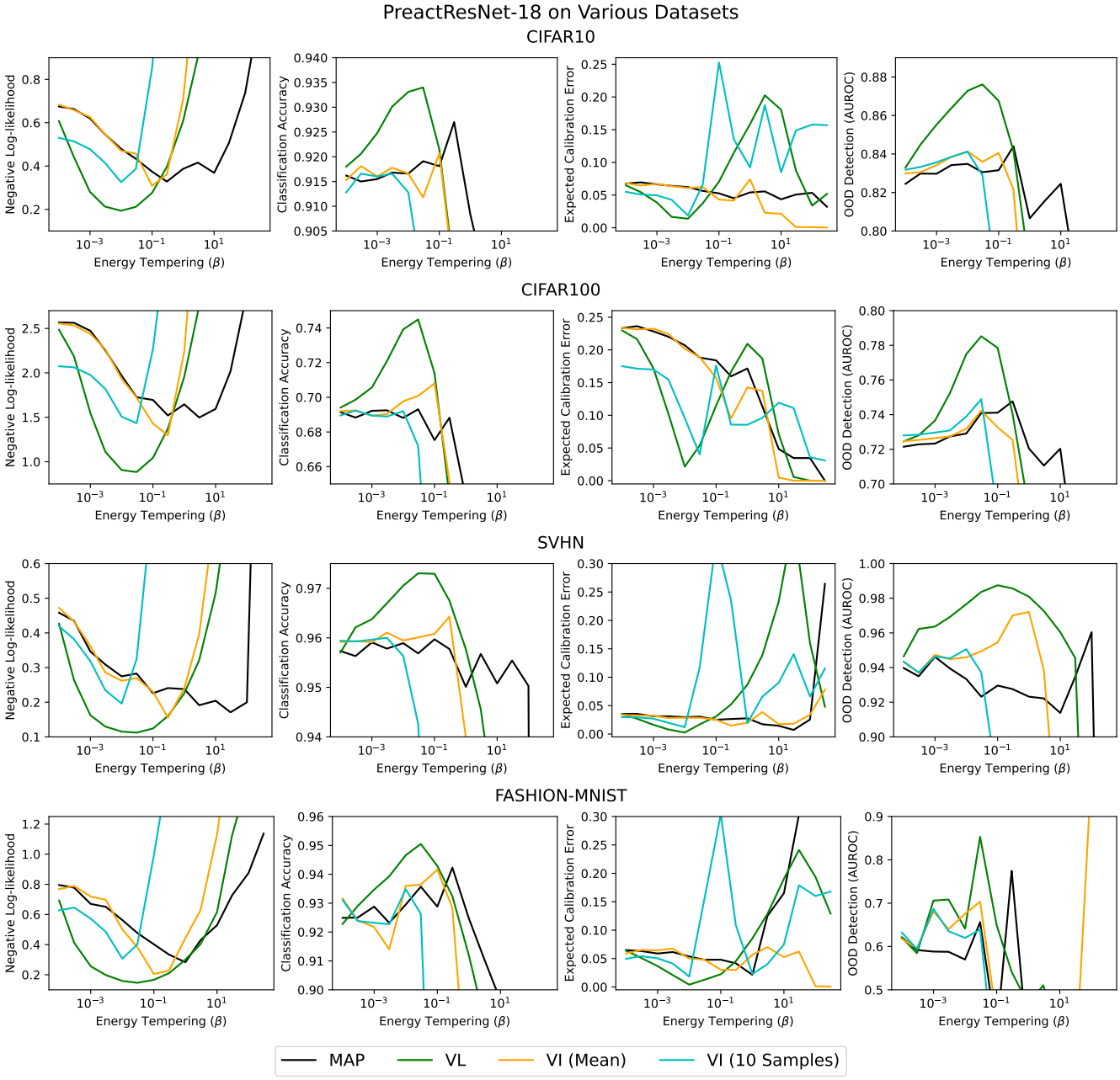

Figure A1. Replication of Figure 1 with a batch size of 64.

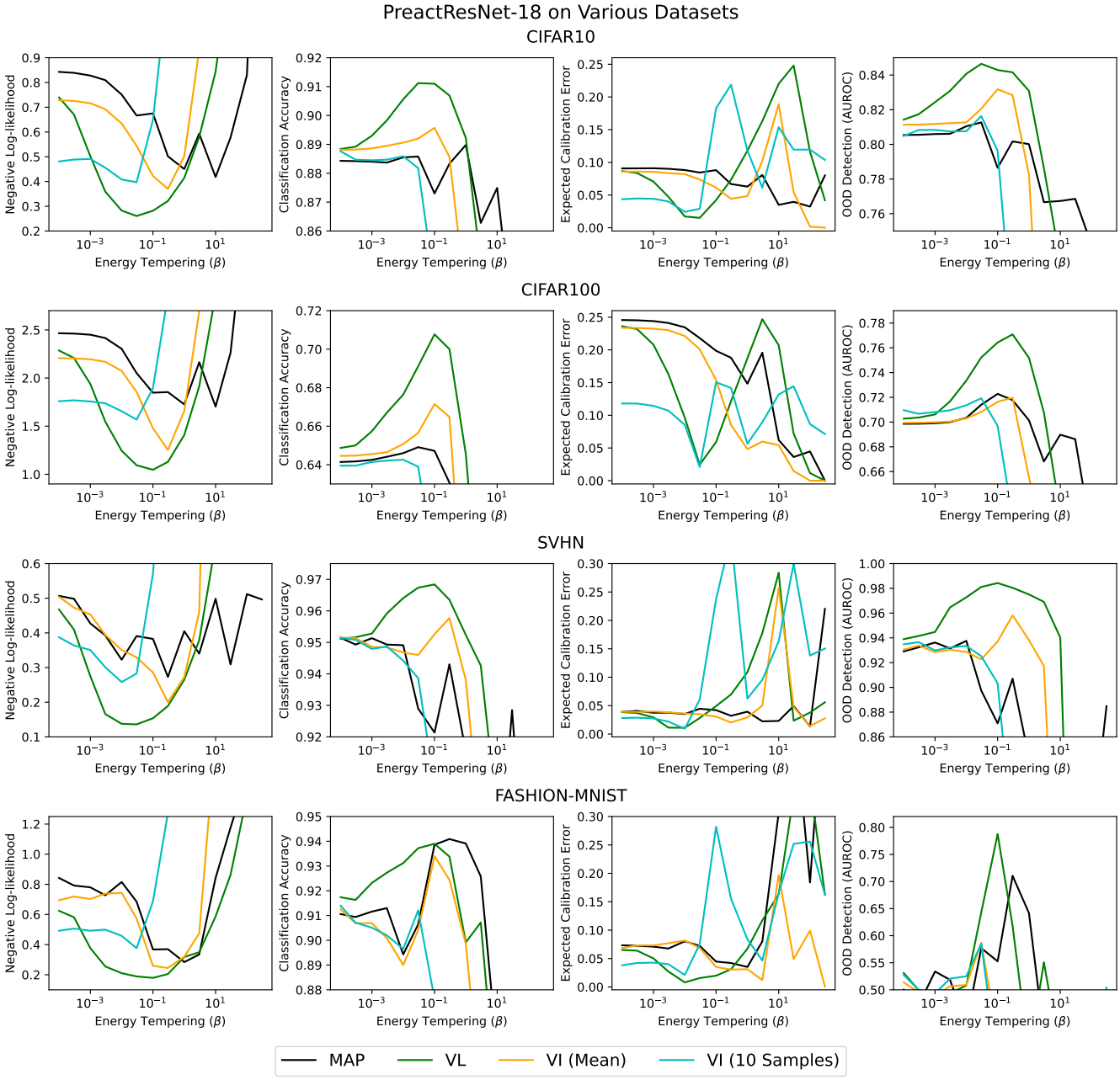

Figure A2. Replication of Figure 1 with a batch size of 256.

References

1. Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; et al. End to end learning for self-driving cars. *arXiv* **2016**, arXiv:1604.07316.
2. Amato, F.; López, A.; Peña-Méndez, E.M.; Vanhara, P.; Hampl, A.; Havel, J. Artificial neural networks in medical diagnosis. *J. Appl. Biomed.* **2013**, *11*, 47–58. [CrossRef]
3. McAllister, R.; Gal, Y.; Kendall, A.; Van Der Wilk, M.; Shah, A.; Cipolla, R.; Weller, A. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017.
4. Welling, M.; Teh, Y.W. Bayesian learning via stochastic gradient Langevin dynamics. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA, 28 June–2 July 2011; pp. 681–688.
5. Azevedo-Filho, A.; Shachter, R.D. Laplace's method approximations for probabilistic inference in belief networks with continuous variables. In *Uncertainty Proceedings 1994*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 28–36.
6. MacKay, D.J. *Information Theory, Inference and Learning Algorithms*; Cambridge University Press: Cambridge, UK, 2003.

7. Ritter, H.; Botev, A.; Barber, D. A scalable laplace approximation for neural networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR 2018), Vancouver, BC, Canada, 30 April–3 May 2018; Volume 6.
8. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight uncertainty in neural networks. *arXiv* **2015**, arXiv:1505.05424.
9. Ober, S.W.; Aitchison, L. Global inducing point variational posteriors for Bayesian neural networks and deep Gaussian processes. *arXiv* **2020**, arXiv:2005.08140.
10. Wainwright, M.J.; Jordan, M.I. *Graphical Models, Exponential Families, and Variational Inference*; Now Publishers Inc.: Delft, The Netherlands, 2008.
11. Neyshabur, B.; Bhojanapalli, S.; McAllester, D.; Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 5947–5956.
12. Barrett, D.G.; Dherin, B. Implicit Gradient Regularization. *arXiv* **2020**, arXiv:2009.11162.
13. Geiping, J.; Goldblum, M.; Pope, P.E.; Moeller, M.; Goldstein, T. Stochastic training is not necessary for generalization. *arXiv* **2021**, arXiv:2109.14119.
14. Hinton, G.E.; Van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In Proceedings of the Sixth Annual Conference on Computational Learning Theory, Santa Cruz, CA, USA, 26–28 July 1993; pp. 5–13.
15. Huang, C.W.; Tan, S.; Lacoste, A.; Courville, A.C. Improving explorability in variational inference with annealed variational objectives. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2018; pp. 9701–9711.
16. Wenzel, F.; Roth, K.; Veeling, B.S.; Swiatkowski, J.; Tran, L.; Mandt, S.; Snoek, J.; Salimans, T.; Jenatton, R.; Nowozin, S. How Good is the Bayes Posterior in Deep Neural Networks Really? *arXiv* **2020**, arXiv:2002.02405.
17. Aitchison, L. A statistical theory of cold posteriors in deep neural networks. *arXiv* **2020**, arXiv:2008.05912.
18. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
19. Rezende, D.J.; Mohamed, S.; Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv* **2014**, arXiv:1401.4082.
20. Friston, K.; Mattout, J.; Trujillo-Barreto, N.; Ashburner, J.; Penny, W. Variational free energy and the Laplace approximation. *NeuroImage* **2007**, *34*, 220–234. [CrossRef]
21. Daunizeau, J.; Friston, K.J.; Kiebel, S.J. Variational Bayesian identification and prediction of stochastic nonlinear dynamic causal models. *Phys. D Nonlinear Phenom.* **2009**, *238*, 2089–2118. [CrossRef]
22. Daunizeau, J. The Variational Laplace approach to approximate Bayesian inference. *arXiv* **2017**, arXiv:1703.02089.
23. Wu, A.; Nowozin, S.; Meeds, E.; Turner, R.E.; Hernández-Lobato, J.M.; Gaunt, A.L. Deterministic variational inference for robust bayesian neural networks. *arXiv* **2018**, arXiv:1810.03958.
24. Haußmann, M.; Hamprecht, F.A.; Kandemir, M. Sampling-free variational inference of bayesian neural networks by variance backpropagation. In Proceedings of the 35th Uncertainty in Artificial Intelligence Conference, Tel Aviv, Israel, 22–25 July 2019; pp. 563–573.
25. MacKay, D.J. A practical Bayesian framework for backpropagation networks. *Neural Comput.* **1992**, *4*, 448–472. [CrossRef]
26. Smith, S.L.; Dherin, B.; Barrett, D.G.; De, S. On the Origin of Implicit Regularization in Stochastic Gradient Descent. *arXiv* **2021**, arXiv:2101.12176.
27. Kunstner, F.; Hennig, P.; Balles, L. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 4156–4167.
28. Khan, M.E.; Immer, A.; Abedi, E.; Korzepa, M. Approximate inference turns deep networks into gaussian processes. *arXiv* **2019**, arXiv:1906.01930.
29. Kuok, S.C.; Yuen, K.V. Broad Bayesian learning (BBL) for nonparametric probabilistic modeling with optimized architecture configuration. *Comput.-Aided Civ. Infrastruct. Eng.* **2021**, *36*, 1270–1287. [CrossRef]
30. Yao, J.; Pan, W.; Ghosh, S.; Doshi-Velez, F. Quality of uncertainty quantification for Bayesian neural network inference. *arXiv* **2019**, arXiv:1906.09686.
31. Guo, C.; Pleiss, G.; Sun, Y.; Weinberger, K.Q. On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Sydney, NSW, Australia, 6–11 August 2017; pp. 1321–1330.
32. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; University of Toronto: Toronto, ON, Canada, 2009.
33. Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A.Y. Reading digits in natural images with unsupervised feature learning. In Proceedings of the NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain, 16–17 December 2011.
34. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity mappings in deep residual networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 630–645.
36. Naeini, M.P.; Cooper, G.; Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
37. Chrabaszcz, P.; Loshchilov, I.; Hutter, F. A Downsampled Variant of ImageNet as an Alternative to the CIFAR datasets. *arXiv* **2017**, arXiv:1707.08819.

38. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
39. Mandt, S.; Hoffman, M.D.; Blei, D.M. Stochastic gradient descent as approximate bayesian inference. *J. Mach. Learn. Res.* **2017**, *18*, 4873–4907.