entropy

Article

Differentiable PAC–Bayes Objectives with Partially Aggregated Neural Networks

Felix Biggs [1] and Benjamin Guedj [1,2,*]

[1] Centre for Artificial Intelligence, Department of Computer Science, University College London, London WC1V 6LJ, UK; fbiggs@cs.ucl.ac.uk
[2] Inria Lille—Nord Europe Research Centre and Inria London, 59800 Lille, France
* Correspondence: benjamin.guedj@inria.fr

Abstract: We make two related contributions motivated by the challenge of training stochastic neural networks, particularly in a PAC–Bayesian setting: (1) we show how averaging over an ensemble of stochastic neural networks enables a new class of partially-aggregated estimators, proving that these lead to unbiased lower-variance output and gradient estimators; (2) we reformulate a PAC–Bayesian bound for signed-*output* networks to derive in combination with the above a directly optimisable, differentiable objective and a generalisation guarantee, without using a surrogate loss or loosening the bound. We show empirically that this leads to competitive generalisation guarantees and compares favourably to other methods for training such networks. Finally, we note that the above leads to a simpler PAC–Bayesian training scheme for sign-*activation* networks than previous work.

Keywords: statistical learning theory; PAC–Bayes theory; deep learning

check for updates

Citation: Biggs, F.; Guedj, B. Differentiable PAC–Bayes Objectives with Partially Aggregated Neural Networks. *Entropy* **2021**, *23*, 1280. https://doi.org/10.3390/e23101280

Academic Editor: Udo Von Toussaint

Received: 22 August 2021
Accepted: 27 September 2021
Published: 29 September 2021

1. Introduction

The use of stochastic neural networks has become widespread in the PAC–Bayesian and Bayesian deep learning [1] literature as a way to quantify predictive uncertainty and obtain generalisation bounds. PAC–Bayesian theorems generally bound the expected loss of *randomised* estimators, so it has proven easier to obtain non-vacuous numerical guarantees on generalisation in such networks.

However, we observe that when training these in the PAC–Bayesian setting, the objective used is generally somewhat divorced from the bound on misclassification loss itself, often because non-differentiability leads to difficulties with direct optimisation. For example, Langford and Caruana [2], Zhou et al. [3], and Dziugaite and Roy [4] all initially train non-stochastic networks before using them as the mode of a distribution, with variance chosen, respectively, through a computationally-expensive sensitivity analysis, as a proportion of weight norms, or by optimising an objective with both a surrogate loss function and a different dependence on the Kullback–Leibler (KL) divergence from their bound.

In exploring methods to circumvent this gap, we also note that PAC–Bayesian bounds can often be straightforwardly adapted to aggregates or averages of estimators, leading directly to analytic and differentiable objective functions (for example, [5]). Unfortunately, averages over deep stochastic networks are usually intractable or, if possible, very costly (as found by [6]).

Motivated by these observations, our main contribution is to obtain a compromise by defining new and general "*partially-aggregated*" Monte Carlo estimators for the average output and gradients of deep stochastic networks (Section 3), with the direct optimisation of PAC–Bayesian bounds in mind. Although our main focus here is on the use of this estimator in a PAC–Bayesian application, we emphasise that the technique applies generally to stochastic networks and thus has links to other variance-reduction techniques for training them, such as the pathwise estimator used in the context of neural networks by [7] amongst many others or Flipout [8]; indeed, it can be used in combination with these techniques. We provide proofs (Section 4) that this application leads to lower variances than a Monte Carlo forward pass and lower variance final-layer gradients than REINFORCE [9].

A further contribution of ours is a first application of this general estimator to non-differentiable "signed-output" networks (with a final output $\in \{-1, +1\}$ and arbitrarily complex other structure, see Section 4). As well as reducing variances as stated above, a small amount of additional structure in combination with partial-aggregation enables us to extend the pathwise estimator to the other layers, which usually requires a fully differentiable network and eases training by reducing the variance of gradient estimates.

We adapt a binary classification bound (Section 5) from Catoni [10] to these networks, yielding straightforward and directly differentiable objectives when used in combination with aggregation. Closing this gap between objectives and bounds leads to improved theoretical properties.

Further, since most of the existing PAC–Bayes bounds for neural networks have a heavy dependency on the distance from initialisation of the obtained solution, we would intuitively expect these lower variances to lead to faster convergence and tighter bounds (from finding low-error solutions nearer to the initialisation). We indeed observe this experimentally, showing that training PAC–Bayesian objectives in combination with partial aggregation leads to competitive experimental generalisation guarantees (Section 6), and improves upon naive Monte Carlo and REINFORCE.

As a useful corollary, this application also leads us to a similar but simpler PAC–Bayesian training method for sign-activation neural networks than Letarte et al. [6], which successfully aggregated networks with all sign activation functions $\in \{+1, -1\}$ and a non-standard tree structure, but incurred an exponential KL divergence penalty and a heavy computational cost (so that in practice they often resorted to a Monte Carlo estimate). Further, the lower variance of our obtained estimator predictions enables us to use the Gibbs estimator directly (where we draw a single sample function for every new example), leading to a modified bound on the misclassification loss which is a factor of two tighter without a significant performance penalty.

We discuss further and outline future work in Section 7.

2. Background

We begin here by setting out our notation and the requisite background.

Generally, we consider parameterised functions, $\{f_\theta : \mathcal{X} \to \mathcal{Y} | \theta \in \Theta \subset \mathbb{R}^N\}$, in a specific form, choosing $\mathcal{X} \subset \mathbb{R}^{d_0}$ and an arbitrary output space $\mathcal{Y}$ which could be for example $\{-1, +1\}$ or $\mathbb{R}$. We wish to find functions minimizing the out-of-sample risk, $R(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\ell(f(\boldsymbol{x}), y)$, for some loss function ℓ, for example the 0-1 misclassification loss for classification, $\ell_{0-1}(y, y') = \mathbf{1}\{y \neq y'\}$, or the binary linear loss, $\ell_{\text{lin}}(y, y') = \frac{1}{2}(1 - yy')$, with $\mathcal{Y} = \{+1, -1\}$. These must be chosen based on an i.i.d. sample $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m \sim \mathcal{D}^m$ from the data distribution $\mathcal{D}$, using the surrogate of in-sample empirical risk, $R_S(f) = \frac{1}{m}\sum_{i=1}^m \ell(f(\boldsymbol{x}_i), y_i)$. We denote the expected and empirical risks under the misclassification and linear losses, respectively R^{0-1}, R^{lin}, R_S^{0-1} and R_S^{lin}.

In this paper, we consider learning a distribution (PAC–Bayesian posterior), Q, over the parameters θ. PAC–Bayesian theorems then provide bounds on the expected generalization risk of randomised classifiers, where every prediction is made using a newly sampled function from our posterior, f_θ, $\theta \sim Q$.

We also consider averaging the above to obtain Q-aggregated prediction functions,

$$F_Q(\boldsymbol{x}) := \mathbb{E}_{\theta\sim Q} f_\theta(\boldsymbol{x}). \tag{1}$$

In the case of a convex loss function, Jensen's inequality lower bounds the risk of the randomised function by its Q-aggregate: $\ell(F_Q(x), y) \leq \mathbb{E}_{f\sim Q}\ell(f(x), y)$. The equality is achieved by the linear loss, a fact we will exploit to obtain an easier PAC–Bayesian optimisation objective in Section 5.

2.1. Analytic Q-Aggregates for Signed Linear Functions

Q-aggregate predictors are analytically tractable for "signed-output" functions (here the sign function and "signed" functions have outputs $\in \{+1, -1\}$, as the terminology "binary", used sometimes in the literature, suggests to us too strongly an output $\in \{0, 1\}$) of the form $f_w(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x})$ under a normal distribution, $Q(w) = N(\boldsymbol{\mu}, \mathbb{I})$, as specifically considered in a PAC–Bayesian context for binary classification by [5], obtaining an differentiable objective similar to the SVM. Provided $\boldsymbol{x} \neq \mathbf{0}$:

$$F_Q(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{w} \sim N(\boldsymbol{\mu}, \mathbb{I})} \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x}) = \operatorname{erf}\left(\frac{\boldsymbol{\mu} \cdot \boldsymbol{x}}{\sqrt{2} \|\boldsymbol{x}\|} \right). \tag{2}$$

In Section 4, we will consider aggregating signed output ($f(x) \in \{+1, -1\}$) functions of a more general form.

2.2. Monte Carlo Estimators for More Complex Q-Aggregates

The framework of Q-aggregates can be extended to less tractable cases (for example, with f_θ a randomised or a "Bayesian" neural network, see, e.g., [1]) through a simple and unbiased Monte Carlo approximation:

$$F_Q(\boldsymbol{x}) = \mathbb{E}_{\theta \sim Q} f_\theta(\boldsymbol{x}) \approx \frac{1}{T} \sum_{t=1}^{T} f_{\theta^t}(\boldsymbol{x}) := \hat{F}_Q(\boldsymbol{x}). \tag{3}$$

If we go on to parameterize our posterior Q by $\phi \in \Phi \subset \mathbb{R}^N$ as Q_ϕ and wish to obtain gradients without a closed form for $F_{Q_\phi}(\boldsymbol{x}) = \mathbb{E}_{\theta \sim Q_\phi} f_\theta(\boldsymbol{x})$, there are two possibilities. One is REINFORCE [9], which requires only a differentiable density function, $q_\phi(\theta)$ and makes a Monte Carlo approximation to the left hand side of the identity $\nabla_\phi \mathbb{E}_{\theta \sim q_\phi} f_\theta(\boldsymbol{x}) = \mathbb{E}_{\theta \sim q_\phi}[f_\theta(\boldsymbol{x}) \nabla_\phi \log q_\phi(\theta)]$.

The other is the pathwise estimator, which additionally requires that $f_\theta(\boldsymbol{x})$ be differentiable w.r.t. θ, and that the probability distribution chosen has a standardization function, S_ϕ, which removes the ϕ dependence, turning a parameterised q_ϕ into a non-parameterised distribution p: for example, $S_{\mu,\sigma}(X) = (X - \mu)/\sigma$ to transform a general normal distribution into a standard normal. If this exists, the right hand side of $\nabla_\phi \mathbb{E}_{\theta \sim q_\phi} f_\theta(\boldsymbol{x}) = \mathbb{E}_{\epsilon \sim p} \nabla_\phi f_{S_\phi^{-1}(\epsilon)}(\boldsymbol{x})$ generally yields lower-variance estimates than REINFORCE (see for a modern survey [11]).

The variance introduced by REINFORCE can make it difficult to train neural networks when the pathwise estimator is not available, for example when non-differentiable activation functions are used. Below we find a compromise between the analytically closed form of (2) and the above estimator that enables us to make differentiable certain classes of network and extend the pathwise estimator where otherwise it could not be used. Through this we are able to stably train a new class of network.

2.3. PAC–Bayesian Approach

We use PAC–Bayes in this paper to obtain generalisation guarantees and theoretically-motivated training methods. The primary bound utilised is based on the following theorem, valid for a loss taking values in $[0, 1]$:

Theorem 1 ([10], Theorem 1.2.6)**.** *Given probability measure P on hypothesis space $\mathcal{F}$ and $\alpha > 1$, for all Q on $\mathcal{F}$ with probability at least $1 - \delta$ over $S \sim \mathcal{D}^m$,*

$$\mathbb{E}_{f \sim Q} R(f) \leq \inf_{\lambda > 1} \Phi^{-1}_{\lambda/m} \left[\mathbb{E}_{f \sim Q} R_S(f) + \frac{\alpha}{\lambda} \Delta \right]$$

with $\Phi^{-1}_\gamma(t) = \frac{1 - \exp(-\gamma t)}{1 - \exp(-\gamma)}$ *and* $\Delta = \mathrm{KL}(Q|P) - \log \delta + 2 \log\left(\frac{\log \alpha^2 \lambda}{\log \alpha} \right)$.

This slightly opaque formulation (used previously by [3]) gives essentially identical results when KL/m is large to the better-known "small-kl" PAC–Bayes bounds originated by Langford and Seeger [12], Seeger et al. [13]. It is chosen because it leads to objectives that are *linear* in the empirical loss and KL divergence, like

$$\mathbb{E}_{f\sim Q}R_S(f) + \frac{\mathrm{KL}(Q|P)}{\lambda}. \tag{4}$$

This objective is minimised by a Gibbs distribution and is closely related to the evidence lower bound (ELBO) usually optimised by Bayesian Neural Networks [1]. Such a connection has been noted throughout the PAC–Bayesian literature; we refer the reader to [14] or [15] for a formalised treatment.

3. The Partial Aggregation Estimator

Here we outline our main contribution: a reformulation of Q-aggregation for neural networks leading to different, lower-variance, Monte Carlo estimators for their outputs and gradients. These estimators apply to networks with a dense final layer, and arbitrary stochastic other structure (for example convolutions, residual layers or a non-feedforward structure). Specifically, they take the form

$$f_\theta(\boldsymbol{x}) = A(\boldsymbol{w}\cdot\boldsymbol{\eta}_{\theta^{\neg w}}(\boldsymbol{x})) \tag{5}$$

with $\theta = \mathrm{vec}(\boldsymbol{w},\theta^{\neg w}) \in \Theta \subset \mathbb{R}^D$, $\boldsymbol{w}\in\mathbb{R}^d$, and $\theta^{\neg w}\in\Theta^{\neg w}\subset\mathbb{R}^{D-d}$ the parameter set excluding $\boldsymbol{w}$, for the non-final layers of the network. These non-final layers are included in $\boldsymbol{\eta}_{\theta^{\neg w}} : \mathcal{X}\to\mathcal{A}^d\subseteq\mathbb{R}^d$ and the final activation is $A:\mathbb{R}\to\mathcal{Y}$. For simplicity we have used a one-dimensional output but we note that the formulation and below derivations trivially extend to a vector-valued function with elementwise activations. We require the distribution over parameters to factorise like $Q(\theta) = Q^{w}(\boldsymbol{w})Q^{\neg w}(\theta^{\neg w})$, which is consistent with the literature.

We recover a similar functional form to that considered in Section 2.1 by rewriting the function as $A(\boldsymbol{w}\cdot\boldsymbol{a})$ with $\boldsymbol{a}\in\mathcal{A}^d$ the randomised hidden-layer activations. The "aggregated" activation function on the final layer, which we define as $I(\boldsymbol{a}) := \int A(\boldsymbol{w}\cdot\boldsymbol{a})\,\mathrm{d}Q^{w}(\boldsymbol{w})$, may then be analytically tractable. For example, with $\boldsymbol{w}\sim N(\boldsymbol{\mu},\mathbb{I})$ and a sign final activation, we recall (2) where $I(\boldsymbol{a}) = \mathrm{erf}\left(\frac{\boldsymbol{\mu}\cdot\boldsymbol{a}}{\sqrt{2}\|\boldsymbol{a}\|}\right)$.

Using these definitions we can write the Q-aggregate in terms of the conditional distribution on the activations, $\boldsymbol{a}$, which takes the form $\tilde{Q}^{\neg w}(\boldsymbol{a}|\boldsymbol{x}) := (\boldsymbol{\eta}_{(\cdot)}(\boldsymbol{x}))\circ Q^{\neg w}$, (i.e., the distribution of $\boldsymbol{\eta}_{\theta^{\neg w}}(\boldsymbol{x})|\boldsymbol{x}$, with $\theta^{\neg w}\sim Q^{\neg w}$). The Q-aggregate can then be stated as

$$\begin{aligned}
F_Q(\boldsymbol{x}) &:= \mathbb{E}_{\theta\sim Q}[f_\theta(\boldsymbol{x})] \\
&= \int_{\theta^{\neg w}}\left[\int_{\mathbb{R}^d} A(\boldsymbol{w}\cdot\boldsymbol{\eta}_{\theta^{\neg w}}(\boldsymbol{x}))\,\mathrm{d}Q^{w}(\boldsymbol{w})\right]\mathrm{d}Q^{\neg w}(\theta^{\neg w}) \\
&= \int_{\theta^{\neg w}} I(\boldsymbol{\eta}_{\theta^{\neg w}}(\boldsymbol{x}))\,\mathrm{d}Q^{\neg w}(\theta^{\neg w}) \\
&= \int_{\mathcal{A}^d} I(\boldsymbol{a})\,\mathrm{d}\{(\boldsymbol{\eta}_{(\cdot)}(\boldsymbol{x}))\circ Q^{\neg w}\}(\boldsymbol{a}) \\
&=: \int_{\mathcal{A}^d} I(\boldsymbol{a})\,\mathrm{d}\tilde{Q}^{\neg w}(\boldsymbol{a}|\boldsymbol{x}).
\end{aligned}$$

In most cases, the final integral cannot be calculated exactly or involves a large summation, so we resort to a Monte Carlo estimate, for each $\boldsymbol{x}$ drawing T samples of the randomised activations, $\{\boldsymbol{a}^t\}_{t=1}^T \sim \tilde{Q}^{\neg w}(\boldsymbol{a}|\boldsymbol{x})$ to obtain the "partially-aggregated" estimator

$$F_Q(\boldsymbol{x}) = \int_{\mathcal{A}^d} I(\boldsymbol{a})\,\mathrm{d}\tilde{Q}^{\neg w}(\boldsymbol{a}|\boldsymbol{x}) \approx \frac{1}{T}\sum_{t=1}^{T} I(\boldsymbol{a}^t) = \hat{F}^*_Q(\boldsymbol{x}). \tag{6}$$

This is quite similar to the original estimator from (3), but in fact the aggregation of the final layer may significantly reduce the variance of the outputs and also make better gradient estimates possible, as we will show below.

3.1. Reduced Variance Estimates

Proposition 1. Lower variance outputs: *For a neural network as defined by Equation (5) and the unbiased Q-aggregation estimators defined by Equations (3) and (6),*

$$\mathbb{V}_Q[\hat{F}^*_Q(\boldsymbol{x})] \le \mathbb{V}_Q[\hat{F}_Q(\boldsymbol{x})].$$

Proof. Treating $\boldsymbol{a}$ as a random variable, always conditioned on $\boldsymbol{x}$, we have

$$\begin{aligned}\mathbb{V}_Q[\hat{F}_Q(\boldsymbol{x})] - \mathbb{V}_Q[\hat{F}^\star_Q(\boldsymbol{x})] &= \mathbb{E}_Q|\hat{F}_Q(\boldsymbol{x})|^2 - \mathbb{E}_Q|\hat{F}^\star_Q(\boldsymbol{x})|^2 \\ &= \frac{1}{T}\mathbb{E}_{\boldsymbol{a}|\boldsymbol{x}}\left[\mathbb{E}_{\boldsymbol{w}}|A(\boldsymbol{w}\cdot\boldsymbol{a})|^2 - |\mathbb{E}_{\boldsymbol{w}}A(\boldsymbol{w}\cdot\boldsymbol{a})|^2\right] \\ &= \frac{1}{T}\mathbb{E}_{\boldsymbol{a}|\boldsymbol{x}}[\mathbb{V}_{\boldsymbol{w}}[A(\boldsymbol{w}\cdot\boldsymbol{a})]] \ge 0.\end{aligned}$$

□

From the above we see that the aggregate outputs estimated through partial-aggregation have lower variances. Next, we consider the two unbiased gradient estimators for the distribution over final-layer weights, $\boldsymbol{w}$, arising from partial-aggregation or REINFORCE (as would be used, for example, where the final layer is non-differentiable). Assuming $Q^{\boldsymbol{w}}$ has a density, $q_\phi(\theta^{\boldsymbol{w}})$, parameterised by ϕ, these use forward samples of $\{\boldsymbol{w}^t, \theta^{\neg \boldsymbol{w},(t)}\}_{t=1}^T$ as:

$$\begin{aligned}\hat{\boldsymbol{G}}(\boldsymbol{x}) &:= \frac{1}{T}\sum_{t=1}^{T} A(\boldsymbol{w}^t\cdot\boldsymbol{\eta}^t)\nabla_\phi \log q_\phi(\boldsymbol{w}^t) \\ \hat{\boldsymbol{G}}^*(\boldsymbol{x}) &:= \frac{1}{T}\sum_{t=1}^{T}\nabla_\phi I_{q_\phi}(\boldsymbol{\eta}^t).\end{aligned}$$

Proposition 2. Lower variance gradients: *Under the conditions of Proposition 1 and the above definitions,*

$$\mathrm{Cov}_Q[\hat{\boldsymbol{G}}^*(\boldsymbol{x})] \preceq \mathrm{Cov}_Q[\hat{\boldsymbol{G}}(\boldsymbol{x})]$$

where $A \preceq B \iff B - A$ is positive semi-definite. Thus, for all $\boldsymbol{u} \neq \boldsymbol{0}$, $\mathbb{V}[\hat{\boldsymbol{G}}^*(\boldsymbol{x})\cdot\boldsymbol{u}] \le \mathbb{V}[\hat{\boldsymbol{G}}(\boldsymbol{x})\cdot\boldsymbol{u}]$.

Proof. Writing $\boldsymbol{v} := \nabla_\phi \log q_\phi(\boldsymbol{w})$ and using the unbiasedness of the estimators,

$$\begin{aligned}&\mathrm{Cov}_Q[\hat{\boldsymbol{G}}_Q(\boldsymbol{x})] - \mathrm{Cov}_Q[\hat{\boldsymbol{G}}^\star_Q(\boldsymbol{x})] \\ &= \mathbb{E}_Q[\hat{\boldsymbol{G}}_Q(\boldsymbol{x})\hat{\boldsymbol{G}}_Q(\boldsymbol{x})^T] - \mathbb{E}_Q[\hat{\boldsymbol{G}}^\star_Q(\boldsymbol{x})\hat{\boldsymbol{G}}^\star_Q(\boldsymbol{x})^T] \\ &= \frac{1}{T}\mathbb{E}_{\boldsymbol{a}|\boldsymbol{x}}\left[\mathbb{E}_{\boldsymbol{w}}[A(\boldsymbol{w}\cdot\boldsymbol{a})^2\boldsymbol{v}\boldsymbol{v}^T] - \nabla_\phi I_{q_\phi}(\boldsymbol{\eta}^t)\left(\nabla_\phi I_{q_\phi}(\boldsymbol{\eta}^t)^T\right)\right] \\ &= \frac{1}{T}\mathbb{E}_{\boldsymbol{a}|\boldsymbol{x}}\left[\mathrm{Cov}_{\boldsymbol{w}}[A(\boldsymbol{w}\cdot\boldsymbol{a})\nabla_\phi \log q_\phi(\boldsymbol{w})]\right] \succeq 0\end{aligned}$$

where in the final line we have used that $\nabla_\phi I_{q_\phi}(\boldsymbol{\eta}^t) = \nabla_\phi \mathbb{E}_{\boldsymbol{w}}[A(\boldsymbol{w}\cdot\boldsymbol{\eta}^t)] = \mathbb{E}_{\boldsymbol{w}}[A(\boldsymbol{w}\cdot\boldsymbol{\eta}^t)\boldsymbol{v}]$. □

3.2. Single Hidden Layer

For clarity (and to introduce notation to be used in Section 4.2) we will briefly consider the case of a neural network with one hidden layer, $f_\theta(\boldsymbol{x}) = A_2(\boldsymbol{w}_2 \cdot \boldsymbol{A}_1(W_1\boldsymbol{x}))$. The randomised parameters are $\theta = \mathrm{vec}(\boldsymbol{w}_2, W_1)$, $W_1 \in \mathbb{R}^{d_1\times d_0}$, $\boldsymbol{w}_2 \in \mathbb{R}^{d_1}$ and the elementwise activations are $\boldsymbol{A}_1 : \mathbb{R}^{d_1} \to \mathcal{A}_1^{d_1} \subseteq \mathbb{R}^{d_1}$ and $A_2 : \mathbb{R} \to \mathcal{Y}$. We choose the distribution $Q(\theta) =: Q_2(\boldsymbol{w}_2)Q_1(W_1)$ to factorise over the layers. This is identical to the above and sets $\boldsymbol{\eta}_{W_1}(\boldsymbol{x}) = \boldsymbol{A}_1(W_1\boldsymbol{x})$.

Sampling $\boldsymbol{a}$ is straightforward if sampling W_1 is. Further, if the final layer aggregate is differentiable, and so is the hidden layer activation $\boldsymbol{A}_1$, we may be able to use the lower-variance pathwise gradient estimator for gradients with respect to Q_1. We note that this may be possible even if the activation A_2 is not differentiable, as in Section 4, where we extend the pathwise estimator where we could not otherwise use it.

Computationally, we may implement the above by analytically finding the distribution on the "pre-activations" $W_1\boldsymbol{x}$ (trivial for the normal distribution) before sampling this and passing through the activation. With the pathwise estimator this is known as the "local reparameterization trick" [16], which can lead to considerable computational savings on parallel minibatches compared to direct hierarchical sampling, $\boldsymbol{a} = A_1(W_1\boldsymbol{x})$ with $W_1 \sim Q_1$. We will utilise this in all our reparameterizable dense networks, and a variation on it to save computation when using REINFORCE in Sections 4.2 and 6.

4. Aggregating Signed-Output Networks

Here we consider a first practical application of the aggregation estimator to stochastic neural networks with a final dense sign-activated layer. We have seen above that this partial aggregation leads to better-behaved training objectives and lower-variance gradient estimates across arbitrary other network structure, It may also allow use of pathwise gradients for the other layers, which would not be possible otherwise due to the non-differentiability of the final layer.

Specifically, these networks take the form of Equation (5) with the final layer activation a sign function and weights drawn from a unit variance normal distribution, $Q^w(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}, \mathbb{I})$. The aggregate $I(\boldsymbol{a})$ is given by Equation (2). Normally-distributed weights are chosen because of the simple analytic forms for the aggregate (reminiscent of the tanh activation occasionally used for neural networks) and KL divergence (effectively an L^2 regularisation penalty); we note however that closed forms are available for other commonly-used distributions such as the Laplace.

Using Equations (3) and (6) with independent samples $\{(\boldsymbol{w}^t, \theta^{\neg \boldsymbol{w},(t)})\}_{t=1}^T \sim Q$ and $\boldsymbol{\eta}^t := \boldsymbol{\eta}_{\theta^{\neg \boldsymbol{w},(t)}}(\boldsymbol{x})$ leads to the two unbiased estimators for the output (henceforth assuming the technical condition $\mathbb{P}_{\boldsymbol{\eta}|\boldsymbol{x}}\{\boldsymbol{\eta} = \boldsymbol{0}\} = 0$ that allows aggregation to be well-defined).

$$\hat{F}_Q(\boldsymbol{x}) := \frac{1}{T}\sum_{t=1}^{T} \operatorname{sign}(\boldsymbol{w}^t \cdot \boldsymbol{\eta}^t) \tag{7}$$

$$\hat{F}_Q^*(\boldsymbol{x}) := \frac{1}{T}\sum_{t=1}^{T} \operatorname{erf}\left(\frac{\boldsymbol{\mu} \cdot \boldsymbol{\eta}^t}{\sqrt{2}\|\boldsymbol{\eta}^t\|}\right). \tag{8}$$

It follows immediately from Propositions 1 and 2 that the latter and the associated gradient estimators have lower variances than the former or the REINFORCE gradient estimates (which we would otherwise have to use due to the non-differentiability of the final layer).

4.1. *Lower Variance Estimates of Aggregated Sign-Output Networks*

We clarify the situation with the lower variance estimates further below. In particular, we find that the reduction in variance from using the partial-aggregation estimator is controlled by the norm $\|\boldsymbol{\mu}\|$, so that for small $\|\boldsymbol{\mu}\|$ (as could be expected early in training) the difference can be large, while as $\|\boldsymbol{\mu}\|$ grows, the difference in variance is controlled and we could reasonably revert to the Monte Carlo (or Gibbs) estimator. Note also that as $F_Q(\boldsymbol{x}) \to \pm 1$ (as expected after training), both variances disappear.

We also show that a stricter condition than Proposition 2 holds on the variances of the aggregated gradients here, and thus that the non-aggregated gradients are noisier in all cases than the aggregate.

Proposition 3. *With the definitions given by Equation (7), for all $\boldsymbol{x} \in \mathbb{R}^{d_0}$, $T \in \mathbb{N}$, and Q with normally-distributed final layer,*

$$0 \leq \mathbb{V}_Q[\hat{F}_Q(\boldsymbol{x})] - \mathbb{V}_Q[\hat{F}^*_Q(\boldsymbol{x})] \leq \frac{1}{T}\left(1 - \left|\operatorname{erf}\left(\frac{\|\boldsymbol{\mu}\|}{\sqrt{2}}\right)\right|^2\right).$$

Proof. The left identity follows directly from Proposition 1. We also have

$$\begin{aligned}\mathbb{V}_Q[\hat{F}_Q(\boldsymbol{x})] - \mathbb{V}_Q[\hat{F}^\star_Q(\boldsymbol{x})] &= \frac{1}{T}\mathbb{E}_{\boldsymbol{a}|\boldsymbol{x}}[\mathbb{V}_{\boldsymbol{w}}[\operatorname{sign}(\boldsymbol{w}\cdot\boldsymbol{a})]] \\ &= \frac{1}{T}\left(1 - \mathbb{E}_{\boldsymbol{a}|\boldsymbol{x}}\left|\operatorname{erf}\left(\frac{\boldsymbol{\mu}\cdot\boldsymbol{\eta}}{\sqrt{2}\|\boldsymbol{\eta}\|}\right)\right|^2\right) \\ &\leq \frac{1}{T}\left(1 - \left|\operatorname{erf}\left(\frac{\|\boldsymbol{\mu}\|}{\sqrt{2}}\right)\right|^2\right).\end{aligned}$$

□

Proposition 4. *Under the conditions of Proposition 3,*

$$\operatorname{Cov}[\hat{\boldsymbol{G}}^*(\boldsymbol{x})] \preceq \operatorname{Cov}[\hat{\boldsymbol{G}}(\boldsymbol{x})] + \frac{1-2/\pi}{T}\mathbb{I}.$$

Thus, for all $\boldsymbol{u}$ with $\|\boldsymbol{u}\| = 1$,

$$\mathbb{V}[\hat{\boldsymbol{G}}^*(\boldsymbol{x})\cdot\boldsymbol{u}] \leq \mathbb{V}[\hat{\boldsymbol{G}}(\boldsymbol{x})\cdot\boldsymbol{u}] + \frac{1-2/\pi}{T}.$$

Proof. It is straightforward to show that

$$\begin{aligned}\hat{\boldsymbol{G}}(\boldsymbol{x}) &:= \frac{1}{T}\sum_{t=1}^{T}\operatorname{sign}(\boldsymbol{w}^t\cdot\boldsymbol{\eta}^t)(\boldsymbol{\mu} - \boldsymbol{w}^t) \\ \hat{\boldsymbol{G}}^*(\boldsymbol{x}) &:= \frac{1}{T}\sum_{t=1}^{T}\frac{\boldsymbol{\eta}^t}{\|\boldsymbol{\eta}^t\|}\sqrt{\frac{2}{\pi}}\exp\left[-\frac{1}{2}\left(\frac{\boldsymbol{\mu}\cdot\boldsymbol{\eta}^t}{\|\boldsymbol{\eta}^t\|}\right)^2\right] \\ \operatorname{Cov}[\hat{\boldsymbol{G}}(\boldsymbol{x})] &= \frac{1}{T}\left(\mathbb{I} - \boldsymbol{G}\boldsymbol{G}^T\right) \\ \operatorname{Cov}[\hat{\boldsymbol{G}}^*(\boldsymbol{x})] &= \frac{1}{T}\left(\mathbb{E}\left[\frac{\boldsymbol{\eta}\boldsymbol{\eta}^T}{\|\boldsymbol{\eta}\|^2}\frac{2}{\pi}e^{-\left(\frac{\boldsymbol{\mu}\cdot\boldsymbol{\eta}}{\|\boldsymbol{\eta}\|}\right)^2}\right] - \boldsymbol{G}\boldsymbol{G}^T\right)\end{aligned}$$

so for $\boldsymbol{u} \neq \boldsymbol{0}$,

$$\begin{aligned}&T\boldsymbol{u}^T\left(\operatorname{Cov}[\hat{\boldsymbol{G}}(\boldsymbol{x})] - \operatorname{Cov}[\hat{\boldsymbol{G}}^*(\boldsymbol{x})]\right)\boldsymbol{u} \\ &= \|\boldsymbol{u}\|^2 - \frac{2}{\pi}\mathbb{E}\left[\frac{|\boldsymbol{u}\cdot\boldsymbol{\eta}|^2}{\|\boldsymbol{\eta}\|^2}e^{-\left(\frac{\boldsymbol{\mu}\cdot\boldsymbol{\eta}}{\|\boldsymbol{\eta}\|}\right)^2}\right] \geq \|\boldsymbol{u}\|^2\left(1 - \frac{2}{\pi}\right) > 0.\end{aligned}$$

Above we have brought $\boldsymbol{u}$ inside the term with an expectation, which is then bounded using Cauchy–Schwarz on $|\boldsymbol{u}\cdot\boldsymbol{\eta}|/\|\boldsymbol{\eta}\| \leq \|u\|$, and $e^{-|t|} \leq 1$ for all $t \in \mathbb{R}$. □

4.2. All Sign Activations

Here we examine an important special case previously examined by Letarte et al. [6]: a feed-forward network with all sign activations and normal weights. This takes the form

$$f_\theta(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}_L \cdot \operatorname{sign}(W_{L-1}\ldots\operatorname{sign}(W_1\boldsymbol{x})\ldots))$$

with $\theta := \operatorname{vec}(\boldsymbol{w}_L, \ldots, W_1)$ and $W_l := [\boldsymbol{w}_{l,1} \ldots \boldsymbol{w}_{l,d_l}]^T$; $l \in \{1, ..., L\}$ are the layer indices. We choose unit-variance normal distributions on the weights, which factorise into $Q_l(W_l) = \prod_{i=1}^{d_l} q_{l,i}(\boldsymbol{w}_{l,i})$ with $q_{l,i} = \mathcal{N}(\boldsymbol{\mu}_{l,i}, \mathbb{I}_{d_{l-1}})$. In the notation of Section 3, $\boldsymbol{\eta}_{\theta^{\neg w}}(\boldsymbol{x}) = \operatorname{sign}(W_{L-1} \ldots \operatorname{sign}(W_1 \boldsymbol{x}) \ldots)$ is the final layer activation, which could easily be sampled by mapping $\boldsymbol{x}$ through the first $L-1$ layers with draws from the weight distribution.

Instead, we go on to make an iterative replacement of the weight distributions on each layer by conditionals on the layer activations to obtain the summation

$$\begin{aligned} F_Q(\boldsymbol{x}) = &\sum_{\boldsymbol{a}_1 \in \{+1,-1\}^{d_1}} \tilde{Q}_1(\boldsymbol{a}_1|\boldsymbol{x}) \quad \times \quad \ldots \\ \ldots \quad \times &\sum_{\boldsymbol{a}_{L-1} \in \{+1,-1\}^{d_{L-1}}} \tilde{Q}_{L-1}(\boldsymbol{a}_{L-1}|\boldsymbol{a}_{L-2}) \operatorname{erf}\left(\frac{\boldsymbol{\mu}_L \cdot \boldsymbol{a}_{L-1}}{\sqrt{2}\|\boldsymbol{a}_{L-1}\|}\right). \end{aligned} \tag{9}$$

The number of terms is exponential in the depth so we instead hierarchically sample the $\boldsymbol{a}_l$. Like local reparameterisation, this leads to a considerable computational saving over sampling a separate weight matrix for every input. The conditionals can be found in closed form: we can factorise individual hidden units $\tilde{Q}_l(\boldsymbol{a}_l|\boldsymbol{a}_{l-1}) := \prod_{i=1}^{d_l} \tilde{q}_{l,i}(a_{l,i}|\boldsymbol{a}_{l-1})$, and find their activation distributions (with $\boldsymbol{a}_0 := \boldsymbol{x}$ and z a dummy variable):

$$\begin{aligned} \tilde{q}_{l,i}(a_{l,i} = \pm 1\,|\boldsymbol{a}_{l-1}) &= \int_0^\infty \mathcal{N}(z; \pm\boldsymbol{\mu}_{l,i} \cdot \boldsymbol{a}_{l-1}, \|\boldsymbol{a}_{l-1}\|^2)\, \mathrm{d}z \\ &= \frac{1}{2}\left[1 \pm \operatorname{erf}\left(\frac{\boldsymbol{\mu}_{l,i} \cdot \boldsymbol{a}_{l-1}}{\sqrt{2}\|\boldsymbol{a}_{l-1}\|}\right)\right]. \end{aligned}$$

A marginalised REINFORCE-style gradient estimator for *conditional* distributions can then be used; this does not necessarily have better statistical properties but in combination with the above is much more computationally efficient. This idea of "conditional sampling" is inspired by the local reinforce trick. Using samples $\{(\boldsymbol{a}_1^t \ldots \boldsymbol{a}_{L-1}^t)\}_{t=1}^T \sim \tilde{Q}$,

$$\frac{\partial F_Q(\boldsymbol{x})}{\partial \boldsymbol{\mu}_{l,i}} \approx \frac{1}{T} \sum_{t=1}^T \operatorname{erf}\left(\frac{\boldsymbol{\mu}_L \cdot \boldsymbol{a}_{L-1}^t}{\sqrt{2}\|\boldsymbol{a}_{L-1}^t\|}\right) \frac{\partial}{\partial \boldsymbol{\mu}_{l,i}} \log \tilde{q}_{l,i}(a_{l,i}^t|\boldsymbol{a}_{l-1}^t). \tag{10}$$

This formulation along with Equation (9) resembles the PBGNet model of [6], but derived in a very different way. Indeed both are equivalent in the single-hidden-layer case, but with more layers PBGNet uses an unusual tree-structured network to make the individual activations independent and avoid an exponential computational dependency on the depth in Equation (9). This makes the above summation exactly calculable but is also still not efficient enough in practice, so they resort further to a Monte Carlo approximation: informally, this draws new samples for every layer l based on an average of those from the previous layer, $\boldsymbol{a}_l|\{\boldsymbol{a}_{l-1}^{(t)}\}_{t=1}^T \sim \frac{1}{T}\sum_{t=1}^T \tilde{Q}(\boldsymbol{a}_l|\boldsymbol{a}_{l-1}^{(t)})$.

This is all justified within the tree-structured framework but leads to an exponential KL penalty which—as hinted by Letarte et al. [6] and shown empirically in Section 6—makes PAC–Bayes bound optimisation strongly favour shallower such networks. Our formulation avoids this, is more general—applying to alternative network structures—and we believe it is significantly easier to understand and use in practice.

5. PAC–Bayesian Objectives with Signed-Outputs

We now move to obtain binary classifiers with guarantees for the expected misclassification error, R^{0-1}, which we do by optimizing PAC–Bayesian bounds. Such bounds (as in Theorem 1) will usually involve the non-differentiable and non-convex misclassification loss ℓ_{0-1}. However, to train a neural network we need to replace this by a differentiable surrogate, as discussed in the introduction.

Here we adopt a different approach by using our signed-output networks, where since $f(x) \in \{+1, -1\}$, there is an exact equivalence between the linear and misclassification losses, $\ell_{0-1}(f(x), y) = \ell_{\text{lin}}(f(x), y)$, avoiding an extra factor of two from the inequality $\ell_{0-1} \leq 2\ell_{\text{lin}}$.

Although we have only moved the non-differentiability into f, the form of a PAC–Bayesian bound and the linearity of the loss and expectation allow us to go further and aggregate,

$$\mathbb{E}_{f\sim Q}\ell_{0-1}(f(\boldsymbol{x}), y) = \mathbb{E}_{f\sim Q}\ell_{\text{lin}}(f(\boldsymbol{x}), y) = \ell_{\text{lin}}(F_Q(\boldsymbol{x}), y) \tag{11}$$

which allows us to use the tools discussed in Section 4 to obtain lower-variance estimates and gradients. Below we prove a small result to show the utility of this:

Proposition 5. *Under the conditions of Proposition 3 and $y \in \{+1, -1\}$,*

$$\begin{aligned}\mathbb{V}_Q[\ell_{\text{lin}}(\hat{F}^*_Q(\boldsymbol{x}), y)] &\leq \mathbb{V}_Q[\ell_{\text{lin}}(\hat{F}_Q(\boldsymbol{x}), y)] \\ &\leq \mathbb{V}_{f\sim Q}[\ell_{0-1}(f(\boldsymbol{x}), y)] = \frac{1}{4}(1 - |F_Q(\boldsymbol{x})|^2).\end{aligned}$$

Proof.

$$\mathbb{V}_Q[\ell_{\text{lin}}(\hat{F}_Q(\boldsymbol{x}), y)] = \mathbb{E}_Q\left|\frac{1}{2}(yF_Q(\boldsymbol{x}) - y\hat{F}_Q(\boldsymbol{x}))\right|^2 = \frac{1}{4}\mathbb{V}_Q[\hat{F}_Q(\boldsymbol{x})]$$

and a similar result for $\hat{F}^*_Q$. $f = \hat{F}_Q$ if $T = 1$ and $\ell_{\text{lin}}(f(\boldsymbol{x}), y) = \ell_{0-1}(f(\boldsymbol{x}), y)$. The result then follows from this and Proposition 3. □

Combining (11) with Theorem 1, we obtain a directly optimizable, differentiable bound on the misclassification loss without introducing the above-mentioned factor of 2.

Theorem 2. *Given P on θ and $\alpha > 1$, for all Q on θ and $\lambda > 1$ simultaneously with probability at least $1 - \delta$ over $S \sim \mathcal{D}^m$,*

$$\mathbb{E}_{\theta\sim Q}R^{0-1}(f_\theta) \leq \Phi^{-1}_{\lambda/m}\left[R^{\text{lin}}_S(F_Q) + \frac{\alpha}{\lambda}\Delta\right]$$

with $\Phi^{-1}_\gamma(t) = \frac{1-\exp(-\gamma t)}{1-\exp(-\gamma)}$, $f_\theta : \mathbb{R}^d \to \{+1, -1\}$, $\theta \in \Theta$, *and* $\Delta = \text{KL}(Q|P) - \log\delta + 2\log\left(\frac{\log\alpha^2\lambda}{\log\alpha}\right)$.

Thus, for each λ, which can be held fixed ("**fix-$\boldsymbol{\lambda}$**") or simultaneously optimized throughout training for automatic regularisation tuning ("**optim-$\boldsymbol{\lambda}$**"), we obtain a gradient descent objective:

$$R^{\text{lin}}_S(\hat{F}^*_Q) + \frac{\text{KL}(Q|P)}{\lambda}. \tag{12}$$

6. Experiments

All experiments (Table 1) run on "binary"-MNIST, dividing MNIST into two classes, of digits 0–4 and 5–9. Neural networks had three hidden layers with 100 units per layer and **sign**, sigmoid (**sgmd**) or **relu** activations, before a single-unit final layer with sign activation. Q was chosen as an isotropic, unit-variance normal distribution with initial means drawn from a truncated normal distribution of variance 0.05. The data-free prior P was fixed equal to the initial Q, as motivated by Dziugaite and Roy [4] (Section 5 and Appendix B).

Table 1. Average (from ten runs) binary-MNIST losses and bounds ($\delta = 0.05$) for the best epoch and optimal hyperparameter settings of various algorithms. Hyperparameters and epochs were chosen by bound if available and non-vacuous, otherwise by training linear loss. Bold numbers indicate the best values and standard deviation is reported in italics.

	mlp	**pbg**	**Reinforce**		**Fix-λ**			**Optim-λ**		
			sign	**relu**	**sign**	**sgmd**	**relu**	**sign**	**sgmd**	**relu**
Train Linear	**0.78**	8.72	26.0	18.6	8.77	7.60	6.35	6.71	6.47	5.41
error, 1σ	*0.08*	*0.08*	*0.8*	*1.4*	*0.04*	*0.19*	*0.10*	*0.11*	*0.18*	*0.16*
Test 0–1	**1.82**	5.26	25.4	17.9	8.73	7.88	6.51	6.85	6.84	5.61
error, 1σ	*0.16*	*0.18*	*1.0*	*1.5*	*0.23*	*0.30*	*0.19*	*0.27*	*0.21*	*0.20*
Bound 0–1	-	40.8	100	100	21.7	18.8	**15.5**	22.6	19.3	16.0
error, 1σ	-	*0.2*	*0.0*	*0.0*	*0.04*	*0.17*	*0.04*	*0.03*	*0.31*	*0.05*

The objectives **fix-λ** and **optim-λ** from Section 5 were used for batch-size 256 gradient descent with Adam [17] for 200 epochs. Every five epochs, the bound (for a minimising λ) was evaluated using the entire training set; the learning rate was then halved if the bound was unimproved from the previous two evaluations. The best hyperparameters were selected using the best bound achieved in these evaluations through a grid search of initial learning rates $\in \{0.1, 0.01, 0.001\}$, sample sizes $T \in \{1, 10, 50, 100\}$. Once these were selected training was repeated 10 times to obtain the values in Table 1.

λ in **optim-λ** was optimised through Theorem 2 on alternate mini-batches with SGD and a fixed learning rate of 10^{-4} (whilst still using the objective (12) to avoid effectively scaling the learning rate with respect to empirical loss by the varying λ). After preliminary experiments in **fix-λ**, we set $\lambda = m = 60{,}000$, the training set size, as is common in Bayesian deep learning.

We also report the values of three baselines: **reinforce**, which uses the fix-λ objective without partial-aggregation, forcing the use of REINFORCE gradients everywhere; **mlp**, an unregularised non-stochastic relu neural network with tanh output activation; and the PBGNet model (**pbg**) from Letarte et al. [6]. For the latter, a misclassification error bound obtained through $\ell_{0-1} \leq 2\ell_{\text{lin}}$ must be used as their test predictions were made through the sign of a prediction function $\in [-1, +1]$, not $\in \{+1, -1\}$. Further, despite significant additional hyperparameter exploration, we were unable to train a three layer network through the PBGNet algorithm directly comparable to our method, likely because of the exponential KL penalty (in their Equation 17) within that framework; to enable comparison, we therefore allowed the number of hidden layers in this scenario to vary $\in \{1, 2, 3\}$. Other baseline tuning and setup was similar to the above, see the Appendix A for more details.

During evaluation **reinforce** draws a new set of weights for every test example, equivalent to the evaluation of the other models; but doing so during training, with multiple parallel samples, is prohibitively expensive. Two different approaches to straightforward, not partially-aggregated, gradient estimation for this case suggest themselves, arising from different approximations to the Q-expected loss of the minibatch, $B \subseteq S$ (with data indices $\mathcal{B}$). From the identities

$$\begin{aligned}\nabla_\phi \mathbb{E}_{\theta\sim q_\phi} R_B(f_\theta) &= \mathbb{E}_{\theta\sim q_\phi} \frac{1}{|B|} \sum_{i\in\mathcal{B}} \ell(f_\theta(\boldsymbol{x}_i), y_i) \nabla_\phi \log q_\phi(\theta) \\ &= \frac{1}{|B|} \sum_{i\in\mathcal{B}} \mathbb{E}_{\theta\sim q_\phi} \ell(f_\theta(\boldsymbol{x}_i), y_i) \nabla_\phi \log q_\phi(\theta)\end{aligned}$$

we obtain two slightly different estimators for $\nabla_\phi \mathbb{E}_{\theta\sim q_\phi} R_B(f_\theta)$:

$$\frac{1}{T|B|} \sum_{t=1}^{T} \sum_{i\in\mathcal{B}} \ell(f_{\theta^{(t,i)}}(\boldsymbol{x}_i), y_i) \nabla_\phi \log q_\phi(\theta^{(t,i)})$$

$$\frac{1}{T|B|}\sum_{i\in\mathcal{B}}\sum_{t=1}^{T}\ell(f_{\theta^t}(\boldsymbol{x}_i),y_i)\nabla_\phi \log q_\phi(\theta^t).$$

The first draws many more samples and has lower variance but is much slower computationally; even aside from the $O(|B|)$ increase in computation, there is a slowdown as the optimised BLAS matrix routines cannot be used, and the very large matrices involved may not fit in memory (see for more information [16]).

Therefore, as is standard in the Bayesian Neural Network literature with the pathwise estimator, we use the latter formulation, which has a similar computational complexity to local-reparameterisation and our marginalised REINFORCE estimator (10). We should note though that in preliminary experiments, the alternate estimator did not appear to lead to improved results. This clarifies the advantages of marginalised sampling, which can lead to lower variance with a similar computational cost.

7. Discussion

The experiments demonstrate that partial-aggregation enables training of multi-layer non-differentiable neural networks in a PAC–Bayesian context, which is not possible with REINFORCE gradients and a multiple-hidden-layer PBGNet [6]. These obtained only vacuous bounds, and our misclassification bounds also improve those of a single-hidden-layer PBGNet.

Our experiments raise a couple of questions: firstly, why is it that lower variance estimates empirically lead to tighter bounds? We speculate that the faster convergence of SGD in this case takes us to a more "local" minimum of the objective, closer to our initialisation. Since most existing PAC–Bayes bounds for neural networks have a very strong dependence on this distance from initialisation through the KL term, this leads to tighter bounds. This distance could also be reduced through other methods we consider out-of-scope, such as the data-dependent bounds employed by Dziugaite and Roy [18] and Letarte et al. [6].

A second and harder question is asking why the non-stochastic mlp model obtains a lower overall error. The bound optimisation is empirically quite conservative, but does not necessarily lead to better generalisation; understanding this gap is a key question in the theory of deep learning.

In future work we will develop significant new tools to extend partial-aggregation to multi-class classification, and to improve test prediction bounds for $\operatorname{sign}(\hat{F}_Q(\boldsymbol{x}))$ with $T > 1$, as in PBGNet, which gave slightly improved predictive performance despite the inferior theoretical guarantees.

Author Contributions: Conceptualization, F.B. and B.G.; Formal analysis, F.B. and B.G.; Methodology, F.B. and B.G.; Project administration, B.G.; Software, F.B.; Writing—original draft, F.B.; Writing—review and editing, F.B. and B.G. Both authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by the U.S. Army Research Laboratory and the U. S. Army Research Office, and by the U.K. Ministry of Defence and the U.K. Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/R013616/1. BG acknowledges partial support from the French National Agency for Research, grants ANR-18-CE40-0016-01 and ANR-18-CE23-0015-02. FB acknowledges the support from the Foundational Artificial Intelligence Centre for Doctoral Training at University College London.

Data Availability Statement: Not applicable

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

Appendix A. Further Experimental Details

Appendix A.1. Aggregating Biases with the Sign Function

We used a bias term in our network layers, leading to a simple extension of the above formulation, omitted in the main text for conciseness:

$$\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{\mu},\Sigma),b\sim\mathcal{N}(\beta,\sigma^2)}\ \operatorname{sign}(\boldsymbol{w}\cdot\boldsymbol{x}+b)=\operatorname{erf}\left(\frac{\boldsymbol{\mu}\cdot\boldsymbol{x}+\beta}{\sqrt{2(\boldsymbol{x}^T\Sigma\boldsymbol{x}+\sigma^2)}}\right)$$

since $\boldsymbol{w}\cdot\boldsymbol{x}+b\sim\mathcal{N}(\boldsymbol{\mu}\cdot\boldsymbol{x}+\beta,\boldsymbol{x}^T\Sigma\boldsymbol{x}+\sigma^2)$ and

$$\begin{aligned}\mathbb{E}_{z\sim\mathcal{N}(\alpha,\beta^2)}\ \operatorname{sign} z &= P(z\geq 0)-P(z<0)\\ &=[1-\Phi(-\alpha/\beta)]-\Phi(-\alpha/\beta)\\ &=2\Phi(\alpha/\beta)-1=\operatorname{erf}(\alpha/\sqrt{2}\beta).\end{aligned}$$

The bias and weight co-variances were chosen to be diagonal with a scale of 1, which leads to some simplification in the above.

Appendix A.2. Dataset Details

We used the MNIST dataset version 3.0.1, available online at http://yann.lecun.com/exdb/mnist/ (accessed on 4 June 2021), which contains 60,000 training examples and 10,000 test examples, which were used without any further split, and rescaled to lie in the range $[0,1]$. For the "binary"-MINST task, the labels $+1$ and -1 were assigned to digits in $\{5,6,7,8,9\}$ and $\{0,1,2,3,4\}$, respectively, and images were scaled into the interval $[0,1]$.

Appendix A.3. Hyperparameter Search for Baselines

The baseline comparison values offered with our experiments were optimized similarly to the above, for completeness we report everything here.

The MLP model had three hidden ReLu layers of size 100 each trained with Adam, a learning rate $\in\{0.1,0.01,0.001\}$ and a batch size of 256 for 100 epochs. Complete test and train evaluation was performed after every epoch, and in the absence of a bound, the model and epoch with lowest train linear loss was selected.

For PBGNet we choose the values of hyperparameters from within these values giving the least bound value. Note that, unlike in [6], we do not allow the hidden size to vary $\{\in 10,50,100\}$, and we use the entire MNIST training set as we do not need a validation set. While attempting to train a three hidden layer network, we also searched through the hyperparameter settings with a batch size of 64 as in the original, but after this failed, we returned to the original batch size of 256 with Adam. All experiments were performed using the code from the original paper, available at https://github.com/gletarte/dichotomize-and-generalize (accessed on 4 June 2021).

Since we were unable to train a multiple-hidden-layer network through the PBGNet algorithm, for this model only we explored different numbers of hidden layers $\in\{1,2,3\}$.

Appendix A.4. Final Hyperparameter Settings

In Table A1 we report the hyperparameter settings used for the experiments in Table 1 after exploration. To save computation, hyperparameter settings that were not learning (defined as having a whole-train-set linear loss of >0.45 after ten epochs) were terminated early. This was also done on the later evaluation runs, where in a few instances the fix-λ sigmoid network failed to train after ten epochs; to handle this we reset the network to obtain the main experimental results.

Table A1. Chosen hyperparameter settings and additional details for results in Table 1. Best hyperparameters were chosen by bound if available and non-vacuous, otherwise by best training linear loss through a grid search as described in Section 6 and Appendix A.3. Run times are rounded to nearest 5 min.

	mlp	**pbg**	**Reinforce**		**Fix-λ**			**Optim-λ**		
			sign	**relu**	**sign**	**relu**	**sgmd**	**sign**	**relu**	**sgmd**
Init. LR	0.001	0.01	0.1	0.1	0.01	0.1	0.1	0.01	0.1	0.1
Samples, T	-	100	100	100	100	50	10	100	100	10
Hid. Layers	3	1	3	3	3	3	3	3	3	3
Hid. Size	100	100	100	100	100	100	100	100	100	100
Mean KL	-	2658	15,020	13,613	2363	3571	3011	5561	3204	4000
Runtime/min	10	5	40	40	35	30	25	35	30	25

For clarity we repeat here the hyperparameter settings and search space:

- Initial Learning Rate $\in \{0.1, 0.01, 0.001\}$.
- Training Samples $\in \{1, 10, 50, 100\}$.
- Hidden Size $= 100$.
- Batch Size $= 256$.
- Fix-λ, $\lambda = m = 60{,}000$.
- Number of Hidden Layers $= 3$ for all models, except PBGNet $\in \{1, 2, 3\}$.

Appendix A.5. Implementation and Runtime

Experiments were implemented using Python and the TensorFlow library [19]. Reported approximate runtimes are for execution on a NVIDIA GeForce RTX 2080 Ti GPU.

References

1. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight Uncertainty in Neural Network. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 37, pp. 1613–1622.
2. Langford, J.; Caruana, R. (Not) Bounding the True Error. In *Advances in Neural Information Processing Systems 14*; Dietterich, T.G., Becker, S., Ghahramani, Z., Eds.; MIT Press: Cambridge, MA, USA, 2002; pp. 809–816.
3. Zhou, W.; Veitch, V.; Austern, M.; Adams, R.P.; Orbanz, P. Non-Vacuous Generalization Bounds at the ImageNet Scale: A PAC-Bayesian Compression Approach. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
4. Dziugaite, G.K.; Roy, D.M. Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. In Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2016, Sydney, NSW, Australia, 11–15 August 2017.
5. Germain, P.; Lacasse, A.; Laviolette, F.; Marchand, M. PAC-Bayesian Learning of Linear Classifiers. In Proceedings of the 26th Annual International Conference on Machine Learning—ICML'09, Montreal, QC, Canada, 14–18 June 2009; pp. 1–8. [CrossRef]
6. Letarte, G.; Germain, P.; Guedj, B.; Laviolette, F. Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., dAlché Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 6872–6882.
7. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. *arXiv* **2013**, arXiv:1312.6114.
8. Wen, Y.; Vicol, P.; Ba, J.; Tran, D.; Grosse, R. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
9. Williams, R.J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
10. Catoni, O. PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning. *IMS Lect. Notes Monogr. Ser.* **2007**, *56*, 1–163. [CrossRef]
11. Mohamed, S.; Rosca, M.; Figurnov, M.; Mnih, A. Monte Carlo Gradient Estimation in Machine Learning. *arXiv* **2019**, arXiv:1906.10652.
12. Langford, J.; Seeger, M. Bounds for Averaging Classifiers. 2001. Available online: https://www.cs.cmu.edu/~jcl/papers/averaging/averaging_tech.pdf (accessed on 4 June 2021)
13. Seeger, M.; Langford, J.; Megiddo, N. An improved predictive accuracy bound for averaging classifiers. In Proceedings of the 18th International Conference on Machine Learning, Williamstown, MA, USA, 28 June–1 July 2001; pp. 290–297.

14. Germain, P.; Bach, F.; Lacoste, A.; Lacoste-Julien, S. PAC-Bayesian Theory Meets Bayesian Inference. In *Advances in Neural Information Processing Systems 29*; Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2016; pp. 1884–1892.
15. Knoblauch, J.; Jewson, J.; Damoulas, T. Generalized Variational Inference: Three Arguments for Deriving New Posteriors. *arXiv* **2019**, arXiv:1904.02063.
16. Kingma, D.P.; Salimans, T.; Welling, M. Variational Dropout and the Local Reparameterization Trick. In *Advances in Neural Information Processing Systems 28*; Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2015; pp. 2575–2583.
17. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
18. Dziugaite, G.K.; Roy, D.M. Data-dependent PAC–Bayes priors via differential privacy. In *Advances in Neural Information Processing Systems 31*; Curran Associates, Inc.: Red Hook, NY, USA, 2018; pp. 8430–8441.
19. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. *arXiv* **2015**, arXiv:1603.04467.