Differentiable PAC-Bayes Objectives with Partially Aggregated Neural Networks

Felix Biggs, Benjamin Guedj

Differentiable PAC-Bayes Objectives with Partially Aggregated Neural Networks

Felix Biggs
Centre for Artificial Intelligence
University College London
United Kingdom
fbiggs@cs.ucl.ac.uk

Benjamin Guedj
Centre for Artificial Intelligence
Inria and University College London
United Kingdom
b.guedj@ucl.ac.uk

Abstract

We make three 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; (2) we show that these lead to provably lower-variance gradient estimates for non-differentiable signed-output networks; (3) we reformulate a PAC-Bayesian bound for these networks to derive a directly optimisable, differentiable objective and a generalisation guarantee, without using a surrogate loss or loosening the bound. This bound is twice as tight as that of Letarte et al. (2019) on a similar network type. We show empirically that these innovations make training easier and lead to competitive guarantees.

1 Introduction

The use of stochastic neural networks has become widespread in the PAC-Bayesian and Bayesian deep learning (Blundell et al., 2015) literature as a way to quantify predictive uncertainty. PAC-Bayesian theorems generally bound the expected loss of *stochastic* estimators, so it has proven easier to obtain non-vacuous numerical guarantees on generalisation in such networks, beginning with Langford and Caruana (2002), picked up by Dziugaite and Roy (2017) in the modern "deep" era, and continued by others including Zhou et al. (2019).

These bounds can often be straightforwardly adapted to aggregates or averages of estimators (as in, for example, Germain et al. (2009)), but averages over deep stochastic networks are generally intractable. Letarte et al. (2019) successfully aggregated networks with sign activation functions $\in \{+1, -1\}$ and a non-standard tree structure, but this incurred an exponential KL divergence penalty, and a heavy computational cost that meant in practice they often resorted to a Monte Carlo estimate.

Our first innovation, not specific to PAC-Bayes, is to find a compromise by defining new "partially-aggregated" Monte Carlo estimators for the average output and gradients of general stochastic networks (Section 3). In the case of non-differentiable "signed-output" networks (with a dense final layer and sign activation, the rest of the network having arbitrarily complex structure) we prove that these have lower variance than REINFORCE (Williams, 1992) and a naive Monte Carlo forward pass (Section 4). This framework additionally leads to a similar but simpler training method for sign-activation neural networks than Letarte et al. (2019).

Next, we observe that when training *neural networks* in the PAC-Bayesian setting, the objective used is generally somewhat divorced from the bound on misclassification loss itself, because non-differentiability leads to difficulties with direct optimisation. For example, Dziugaite and Roy (2017) use an objective with both a surrogate loss function and a different dependence on the KL from their bound, and Letarte et al. (2019) use a different loss function which leads to a bound on the

Preprint. Under review.

misclassification loss which is a factor of two worse. This contrasts with the situation for other estimator classes, where the bound may directly lead to a differentiable objective (for example in Germain et al. (2009)).

Motivated by this, we show (Section 5) that a straightforward adaptation of a bound from Catoni (2007) to our signed-output networks, in combination with aggregation, yields straightforward and directly differentiable objectives: one that fixes a regularisation parameter, and another that tunes it automatically through the bound. This closes the previous gap between bounds and objectives in neural networks. We demonstrate that training these objectives in combination with our partial aggregation estimators leads to competitive experimental generalisation guarantees (Section 6). A brief discussion follows in Section 7.

2 Background

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

The standard statistical learning perspective considers parameterised functions, $\{f_\theta : \mathscr{X} \to \mathscr{Y} | \theta \in \Theta\}$, in a specific form, choosing $\mathscr{X} \subset \mathbb{R}^{d_0}$ and an arbitrary output space $\mathscr{Y}$ which could be for example $\{-1,+1\}$ or $\mathbb{R}$. In general, we wish find functions minimizing the out-of-sample expected risk, $R(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathscr{D}}\ell(f(\mathbf{x}),y)$, 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 $\mathscr{Y} = \{+1,-1\}$. These must be chosen based on an i.i.d. sample $S = \{(\mathbf{x}_i,y_i)\}_{i=1}^m \sim \mathscr{D}^m$ from the data distribution $\mathscr{D}$, using the surrogate of in-sample empirical risk, $R_S(f) = \frac{1}{m}\sum_{i=1}^m \ell(f(\mathbf{x}_i),y_i)$. We denote the expected and empirical risks under the misclassification and linear losses respectively $R^{0-1}, R^{\text{lin}}, R_S^{0-1}$ 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 stochastic 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,

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

In the case of a convex loss function, Jensen's inequality lower bounds the risk of the stochastic function by its Q-aggregate: $\ell(F_Q(x),y) \leq \mathbb{E}_{f\sim Q}\ell(f(x),y)$. This is an equality for the linear loss, which 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[1] of the form $f_w(\mathbf{x}) = \text{sign}(\mathbf{w}\cdot\mathbf{x})$ under a normal distribution, $Q(w) = N(\mu,\mathbb{I})$, as first considered in a PAC-Bayesian context for binary classification by Germain et al. (2009), obtaining an differentiable objective similar to the SVM. Provided $\mathbf{x} \neq \mathbf{0}$:

$$F_Q(\mathbf{x}) := \mathbb{E}_{\mathbf{w}\sim N(\mu,\mathbb{I})}\text{sign}(\mathbf{w}\cdot\mathbf{x}) = \text{erf}\left(\frac{\mu\cdot\mathbf{x}}{\sqrt{2}\|\mathbf{x}\|}\right). \tag{1}$$

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

[1] Here the sign function and "signed" functions have outputs $\in \{+1,-1\}$, as the terminology "binary", as used in Letarte et al. (2019), suggests to us too strongly an output $\in \{0,1\}$.

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 stochastic or "Bayesian" neural network, see Blundell et al. (2015)) through a simple Monte Carlo approximation:

$$F_Q(\mathbf{x}) = \mathbb{E}_{\theta\sim Q} f_\theta(\mathbf{x}) \approx \frac{1}{T}\sum_{t=1}^{T} f_{\theta^t}(\mathbf{x}) := \hat{F}_Q(\mathbf{x}). \tag{2}$$

If we go on to parameterize our posterior Q by ϕ as Q_ϕ and wish to obtain gradients without a closed form for $F_{Q_\phi}(\mathbf{x}) = \mathbb{E}_{\theta\sim Q_\phi} f_\theta(\mathbf{x})$, there are two possibilities. One is REINFORCE (Williams, 1992), which requires only a differentiable density function, $q_\phi(\theta)$ and makes a Monte Carlo approximation to the left hand side of the identity $\nabla_\phi \mathbb{E}_{\theta\sim q_\phi} f_\theta(\mathbf{x}) = \mathbb{E}_{\theta\sim q_\phi}[f_\theta(\mathbf{x})\nabla_\phi \log q_\phi(\theta)]$.

The other is the pathwise estimator (used in the context of neural networks by Kingma and Welling, 2013, amongst others), which additionally requires that $f_\theta(\mathbf{x})$ be differentiable w.r.t. θ, and that the probability distribution chosen has a standardization function, S_ϕ, which removes the ϕ dependence: for example, $S_{\mu,\sigma}(X) = (X-\mu)/\sigma$ for a normal distribution. If this exists, the right hand side of $\nabla_\phi \mathbb{E}_{\theta\sim q_\phi} f_\theta(\mathbf{x}) = \mathbb{E}_{\varepsilon\sim p} \nabla_\phi f_{S_\phi^{-1}(\varepsilon)}(\mathbf{x})$ generally yields lower-variance estimates than REINFORCE (see Mohamed et al., 2019, for a modern survey).

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 (1) 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. *(Catoni, 2007, Theorem 1.2.6) Given P on $\mathscr{F}$ and $\alpha > 1$, for all Q on $\mathscr{F}$ with probability at least $1-\delta$ over $S \sim \mathscr{D}^m$,*

$$\mathbb{E}_{f\sim Q} R(f) \leq \inf_{\lambda>1} \Phi^{-1}_{\lambda/m}\left[\mathbb{E}_{f\sim Q} R_S(f) + \frac{\alpha}{\lambda}\left(\mathrm{KL}\left(Q|P\right) - \log\delta + 2\log\left(\frac{\log\alpha^2\lambda}{\log\alpha}\right)\right)\right]$$

with $\Phi_\gamma^{-1}(t) = \frac{1-\exp(-\gamma t)}{1-\exp(-\gamma)}$.

This slightly opaque formulation (used previously by Zhou et al., 2019) gives essentially identical results when KL/m is large to the better-known "small-kl" bounds originated by Langford and Seeger (2001); Seeger et al. (2001). It is chosen because it leads to objectives that are *linear* in the empirical loss and KL divergence, like

$$\mathbb{E}_{f\sim Q} R_S(f) + \frac{\mathrm{KL}\left(Q|P\right)}{\lambda}. \tag{3}$$

This objective is minimised by a Gibbs Posterior and is closely related to the evidence lower bound (ELBO) usually optimised by Bayesian Neural Networks (Blundell et al., 2015). Such a connection has been noted throughout the PAC-Bayesian literature; we refer the reader to Germain et al. (2016) or Knoblauch et al. (2019) for a formalised treatment.

3 Partial Aggregation of Stochastic Neural Networks

Here we consider a reformulation of Q-aggregation for a general class of neural networks that leads to a different Monte Carlo estimator for their outputs and gradients. These networks are those with a dense final layer, taking the form

$$f_\theta(\mathbf{x}) = A(\mathbf{w} \cdot \eta_{\theta^{\neg\mathbf{w}}}(\mathbf{x}))$$

with $\theta = \text{vec}(\mathbf{w}, \theta^{\neg\mathbf{w}}) \in \Theta$, $\mathbf{w} \in \mathbb{R}^d$, $\eta_{\theta^{\neg\mathbf{w}}} : \mathscr{X} \to \mathscr{A}^d \subseteq \mathbb{R}^d$ and $A : \mathbb{R} \to \mathscr{Y}$. $\theta^{\neg\mathbf{w}} \in \Theta^{\neg\mathbf{w}}$ is the parameter set excluding $\mathbf{w}$, for the non-final layers of the network. 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 $Q(\theta) = Q^{\mathbf{w}}(\mathbf{w})Q^{\neg\mathbf{w}}(\theta^{\neg\mathbf{w}})$, which is consistent with the literature.

We recover a similar functional form to that considered in Section 2.1 by rewriting the function as $A(\mathbf{w} \cdot \mathbf{a})$ with $\mathbf{a} \in \mathscr{A}^d$ the hidden-layer activations. The "aggregated" activation function on the final layer, defined as $I(\mathbf{a}) := \int A(\mathbf{w} \cdot \mathbf{a}) \, \mathrm{d}Q^{\mathbf{w}}(\mathbf{w})$, may then be analytically tractable, as in (1), where with $\mathbf{w} \sim N(\mu, \mathbb{I})$ and a sign final activation, $I(\mathbf{a}) = \text{erf}\left(\frac{\mu \cdot \mathbf{a}}{\sqrt{2}\|\mathbf{a}\|}\right)$.

With these definitions we can write the Q-aggregate in terms of the conditional distribution on the activations, $\mathbf{a}$, which is a push-forward measure, $\tilde{Q}^{\neg\mathbf{w}}(\mathbf{a}|\mathbf{x}) := (\eta_{(\cdot)}(\mathbf{x})) \circ Q^{\neg\mathbf{w}}$, (i.e. the distribution of $\eta_{\theta^{\neg\mathbf{w}}}(\mathbf{x})|\mathbf{x}$, with $\theta^{\neg\mathbf{w}} \sim Q^{\neg\mathbf{w}}$) as

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

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 $\mathbf{x}$ drawing T samples of the activations, $\{\mathbf{a}^t\}_{t=1}^T \sim \tilde{Q}^{\neg\mathbf{w}}(\mathbf{a}|\mathbf{x})$ to obtain the "partially-aggregated" estimator

$$F_Q(\mathbf{x}) = \int_{\mathscr{A}^d} I(\mathbf{a}) \, \mathrm{d}\tilde{Q}^{\neg\mathbf{w}}(\mathbf{a}|\mathbf{x}) \approx \frac{1}{T} \sum_{t=1}^{T} I(\mathbf{a}^t) = \hat{F}_Q^*(\mathbf{x}). \tag{4}$$

This is quite similar to the original estimator from (2), 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 in the case of "signed"-output networks.

3.1 Single Hidden Layer

For clarity (and to introduce notation used in Section 4.3) we will briefly consider the case of a neural network with one hidden layer, $f_\theta(\mathbf{x}) = A_2(\mathbf{w}_2 \cdot \mathbf{A}_1(W_1\mathbf{x}))$, with parameters $\theta = \text{vec}(\mathbf{w}_2, W_1)$, $W_1 \in \mathbb{R}^{d_1 \times d_0}$, $\mathbf{w}_2 \in \mathbb{R}^{d_1}$ and elementwise activations $\mathbf{A}_1 : \mathbb{R}^{d_1} \to \mathscr{A}_1^{d_1} \subseteq \mathbb{R}^{d_1}$ and $A_2 : \mathbb{R} \to \mathscr{Y}$. The distribution $Q(\theta) =: Q_2(\mathbf{w}_2)Q_1(W_1)$ factorises over the two layers. This is identical to the above and sets $\eta_{W_1}(\mathbf{x}) = \mathbf{A}_1(W_1\mathbf{x})$.

Sampling $\mathbf{a}$ is straightforward; one method involves analytically finding the distribution on the "pre-activations", a trivial operation for the normal distribution among others, before sampling this and passing through the activation. In combination with the pathwise gradient estimator, this is known as the "local reparameterization trick" (Kingma et al., 2015), and can lead to considerable computational savings on parallel minibatches compared to direct hierarchical sampling, $\mathbf{a} = A_1(W_1\mathbf{x})$ with $W_1 \sim Q_1$. We will utilise this in all our reparameterizable dense networks, and a variation on it in Section 4.3.

4 Aggregating Signed-Output Neural Networks

Here we consider multi-layer feed-forward networks with a final sign activation function and unit variance normal distribution for the final layer, $Q^{\mathbf{w}}(\mathbf{w}) = \mathcal{N}(\mu, \mathbb{I})$, so the aggregate is given by (1). Using (2) and (4) with independent samples $\{(\mathbf{w}^t, \theta^{\neg \mathbf{w},(t)})\}_{t=1}^T \sim Q$ and $\eta^t := \eta_{\theta^{\neg \mathbf{w},(t)}}(\mathbf{x})$ leads to the two estimators[2]

$$\hat{F}_Q(\mathbf{x}) := \frac{1}{T}\sum_{t=1}^{T} \operatorname{sign}(\mathbf{w}^t \cdot \eta^t) \qquad \hat{F}^*_Q(\mathbf{x}) := \frac{1}{T}\sum_{t=1}^{T} \operatorname{erf}\left(\frac{\mu \cdot \eta^t}{\sqrt{2}\|\eta^t\|}\right). \tag{5}$$

Below we discuss how the second estimator, using aggregation of the final layer, leads to better-behaved training objectives and lower-variance gradient estimates, enabling stable training of a previously difficult network type.

4.1 Lower Variance Estimates of Aggregated Sign-Output Networks

Proposition 1. *With the definitions given in (5), for all $\mathbf{x} \in \mathbb{R}^{d_0}$, $T \in \mathbb{N}$, and Q with normally-distributed final layer,*

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

Proof. Treating η as a random variable, always conditioned on $\mathbf{x}$,

$$\mathbb{V}_Q[\hat{F}_Q(\mathbf{x})] = \frac{1}{T}\mathbb{V}[\operatorname{sign}(\mathbf{w}\cdot\eta)] = \frac{1}{T}(1 - |F_Q(\mathbf{x})|^2)$$
$$\mathbb{V}_Q[\hat{F}^*_Q(\mathbf{x})] = \frac{1}{T}\mathbb{V}\left[\operatorname{erf}\left(\frac{\mu\cdot\eta}{\sqrt{2}\|\eta\|}\right)\right] = \frac{1}{T}\left(\mathbb{E}_\eta\left|\operatorname{erf}\left(\frac{\mu\cdot\eta}{\sqrt{2}\|\eta\|}\right)\right|^2 - |F_Q(\mathbf{x})|^2\right).$$

The result follows as the error function is point-wise smaller than 1. □

Proposition 2. *Under the conditions of Proposition 1 and $y \in \{+1, -1\}$,*

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

Proof.

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

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

Note that as $F_Q(\mathbf{x}) \to \pm 1$, both variances disappear, and that the difference in variances of $\hat{F}^*_Q$ and $\hat{F}_Q$ will be biggest in the regime $\|\mu\| \ll 1$.

4.2 Lower Variance Gradient Estimators

Final layer derivative estimators given through REINFORCE and aggregation are derived from the two restatements below:

$$\mathbf{G}(\mathbf{x}) := \frac{\partial}{\partial \mu} F_Q(\mathbf{x}) = \mathbb{E}_{\mathbf{w}}\mathbb{E}_{\eta|\mathbf{x}} \operatorname{sign}(\mathbf{w}\cdot\eta)(\mu - \mathbf{w}) = \mathbb{E}_{\eta|\mathbf{x}} \frac{\eta}{\|\eta\|}\sqrt{\frac{2}{\pi}} \exp\left[-\frac{1}{2}\left(\frac{\mu\cdot\eta}{\|\eta\|}\right)^2\right].$$

[2] Henceforth assuming the technical condition $\mathbb{P}_{\eta|\mathbf{x}}\{\eta = \mathbf{0}\} = 0$ that allows aggregation to be well-defined.

This gives rise to the gradient estimators (using the samples from (5)):

$$\hat{\mathbf{G}}(\mathbf{x}) := \frac{1}{T}\sum_{t=1}^{T} \operatorname{sign}(\mathbf{w}^t \cdot \eta^t)(\mu - \mathbf{w}^t) \qquad \hat{\mathbf{G}}^*(\mathbf{x}) := \frac{1}{T}\sum_{t=1}^{T} \frac{\eta^t}{\|\eta^t\|}\sqrt{\frac{2}{\pi}} \exp\left[-\frac{1}{2}\left(\frac{\mu \cdot \eta^t}{\|\eta^t\|}\right)^2\right]$$

Proposition 3. *Under the conditions of Proposition 1 and with these definitions,*

$$\operatorname{Cov}[\hat{\mathbf{G}}^*(\mathbf{x})] \prec \operatorname{Cov}[\hat{\mathbf{G}}(\mathbf{x})]$$

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

Proof. It is straightforward to show that

$$\operatorname{Cov}[\hat{\mathbf{G}}(\mathbf{x})] = \frac{1}{T}\left(\mathbb{I} - \mathbf{G}\mathbf{G}^T\right) \qquad \operatorname{Cov}[\hat{\mathbf{G}}^*(\mathbf{x})] = \frac{1}{T}\left(\mathbb{E}\left[\frac{\eta\eta^T}{\|\eta\|^2}\frac{2}{\pi}e^{-\left(\frac{\mu\cdot\eta}{\|\eta\|}\right)^2}\right] - \mathbf{G}\mathbf{G}^T\right)$$

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

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

The first equality follows directly from the linearity of the expectation. □

If the rest of the layers are reparameterizable, we can also go on to use the pathwise estimator to estimate gradients in the aggregated case, which is not possible otherwise. We show experimentally this is significantly easier. Firstly, though, we consider an important special case where this is not true: a feed forward network with all sign activations.

4.3 All Sign Activations

Choosing all sign activations and unit-variance normal distributions on the weights of each feed-forward layer,

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

with $\theta := \operatorname{vec}(\mathbf{w}_L, \dots, W_1)$ and $W_l := [\mathbf{w}_{l,1} \dots \mathbf{w}_{l,d_l}]^T$; $l \in \{1, ..., L\}$ indexes layers. The distribution factorises into $Q_l(W_l) = \prod_{i=1}^{d_l} q_{l,i}(\mathbf{w}_{l,i})$ with $q_{l,i} = \mathcal{N}(\mu_{l,i}, \mathbb{I}_{d_{l-1}})$.

In the notation of Section 3, $\eta_{\theta^{-\mathbf{w}}}(\mathbf{x}) = \operatorname{sign}(W_{L-1} \dots \operatorname{sign}(W_1\mathbf{x}) \dots)$ is the final layer activation, which could easily be sampled by mapping $\mathbf{x}$ 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

$$F_Q(\mathbf{x}) = \sum_{\mathbf{a}_1 \in \{+1,-1\}^{d_1}} \cdots \sum_{\mathbf{a}_{L-1} \in \{+1,-1\}^{d_{L-1}}} \operatorname{erf}\left(\frac{\mu_L \cdot \mathbf{a}_{L-1}}{\sqrt{2}\|\mathbf{a}_{L-1}\|}\right) \tilde{Q}_{L-1}(\mathbf{a}_{L-1}|\mathbf{a}_{L-2}) \dots \tilde{Q}_1(\mathbf{a}_1|\mathbf{x}) \tag{6}$$

and hierarchically sample the $\mathbf{a}_l$; 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 as $\tilde{Q}_l(\mathbf{a}_l|\mathbf{a}_{l-1}) := \prod_{i=1}^{d_l} \tilde{q}_{l,i}(a_{l,i}|\mathbf{a}_{l-1})$, and (with $\mathbf{a}_0 := \mathbf{x}$)

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

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. Using samples $\{(\mathbf{a}_1^t \dots \mathbf{a}_{L-1}^t)\}_{t=1}^T \sim \tilde{Q}$,

$$\frac{\partial F_Q(\mathbf{x})}{\partial \mu_{l,i}} \approx \frac{1}{T}\sum_{t=1}^{T} \operatorname{erf}\left(\frac{\mu_L \cdot \mathbf{a}_{L-1}^t}{\sqrt{2}\|\mathbf{a}_{L-1}^t\|}\right) \frac{\partial}{\partial \mu_{l,i}} \log \tilde{q}_{l,i}(a_{l,i}^t|\mathbf{a}_{l-1}^t). \tag{7}$$

The above formulation somewhat resembles the PBGNet model of Letarte et al. (2019), but derived in a very different way. Both are equivalent in the single-hidden-layer case, but with more layers PBGNet uses an unusual tree-structured network for which the exact aggregate can be calculated (very expensively, though avoiding an exponential dependency on depth). Generally, though, to avoid this cost, they resort to a Monte Carlo approximation: informally, this draws new samples for every layer l based on an average of those from the previous layer, $\mathbf{a}_l|\{\mathbf{a}_{l-1}^{(t)}\}_{t=1}^T \sim \frac{1}{T}\sum_{t=1}^T \tilde{Q}(\mathbf{a}_l|\mathbf{a}_{l-1}^{(t)})$.

This is all justified within the tree-structured framework but leads to an exponential KL penalty which—as hinted by Letarte et al. (2019) and shown empirically in Section 6—makes PAC-Bayes bound optimisation strongly favour shallower such networks. In addition to this practical drawback, our formulation is more general and we claim it makes the fundamental ideas of partial-aggregation and marginalised sampling significantly clearer.

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 $f(\mathbf{x}) \in \{+1,-1\}$, there is an exact equivalence between the linear and misclassification losses, $\ell_{0-1}(f(x),y) = \ell_{\text{lin}}(f(x),y)$, avoiding the extra factor of two from the inequality $\ell_{0-1} \leq 2\ell_{\text{lin}}$ used by Letarte et al. (2019, Section 2.1).

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,

$$\mathbb{E}_{f\sim Q}\ell_{0-1}(f(\mathbf{x}),y') = \mathbb{E}_{f\sim Q}\ell_{\text{lin}}(f(x),y) = \ell_{\text{lin}}(F_Q(\mathbf{x}),y') \tag{8}$$

which as we saw in Section 2.1 can make some such non-differentiable functions differentiable.

Combining (8) with Theorem 1, we obtain a directly optimizable, differentiable bound on the misclassification loss without introducing an extra 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 $S \sim \mathscr{D}^m$,*

$$\mathbb{E}_{\theta\sim Q}R^{0-1}(f_\theta) \leq \Phi^{-1}_{\lambda/m}\left[R_S^{\text{lin}}(F_Q) + \frac{\alpha}{\lambda}\left(\text{KL}\left(Q|P\right) - \log\delta + 2\log\left(\frac{\log\alpha^2\lambda}{\log\alpha}\right)\right)\right]$$

with $\Phi_\gamma^{-1}(t) = \frac{1-\exp(-\gamma t)}{1-\exp(-\gamma)}$ and $f_\theta : \mathbb{R}^d \to \{+1,-1\}$, $\theta \in \theta$.

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

$$R_S^{\text{lin}}(\hat{F}_Q^*) + \frac{\text{KL}(Q|P)}{\lambda}. \tag{9}$$

6 Experiments

All experiments 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 (2017, Section 5 and Appendix B).

The objectives **fix-λ** and **optim-λ** from Section 5 were used for batch-size 256 gradient descent with Adam (Kingma and Ba, 2014) for 200 epochs. Every five epochs, the bound (for a minimising λ) was

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.

	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*

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 (9) to avoid effectively scaling the learning rate with respect to empirical loss by the varying λ). After preliminary experiments in **fix-λ**, we set $\lambda = m = 60000$, 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. (2019), with the misclassification error bound obtained through $\ell_{0-1} \leq 2\ell_{\text{lin}}$. Despite significant additional hyperparameter exploration for the latter, 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 for more details.

7 Discussion

The experiments demonstrate that partial-aggregation enables training of multi-layer non-differentiable neural networks in a PAC-Bayesian context: REINFORCE gradients and a multiple-hidden-layer PBGNet (Letarte et al., 2019) obtained only non-vacuous bounds, and our misclassification bounds improve those of a single-hidden-layer PBGNet. We note that our bound optimisation is empirically quite conservative, and the non-stochastic mlp model obtains a lower overall error; understanding this gap is one of the key questions in the theory of deep learning. Finally, we also observe that using $\text{sign}(\hat{F}_Q(\mathbf{x}))$ with $T > 1$ for test prediction, as in PBGNet, gave improved empirical results despite the inferior theoretical guarantees; we consider this an interesting avenue of future research.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015. URL `https://www.tensorflow.org/`.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622. PMLR, 2015. URL http://proceedings.mlr.press/v37/blundell15.html.

Olivier Catoni. Pac-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning. *IMS Lecture Notes Monogr. Ser.*, 56:1–163, 2007. ISSN 0749-2170. doi: 10.1214/074921707000000391.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *Conference on Uncertainty in Artificial Intelligence 33.*, 2017.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8. ACM Press, 2009. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553419.

Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. PAC-Bayesian theory meets bayesian inference. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1884–1892. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6569-pac-bayesian-theory-meets-bayesian-inference.pdf.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick.pdf.

Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. Generalized Variational Inference: Three arguments for deriving new Posteriors. *arXiv preprint arXiv:1904.02063*, 2019.

John Langford and Rich Caruana. (Not) Bounding the True Error. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 809–816. MIT Press, 2002. URL http://papers.nips.cc/paper/1968-not-bounding-the-true-error.pdf.

John Langford and Matthias Seeger. Bounds for averaging classifiers. 2001. URL http://www.cs.cmu.edu/~jcl/papers/averaging/averaging_tech.pdf.

Gaël Letarte, Pascal Germain, Benjamin Guedj, and Francois Laviolette. Dichotomize and generalize: PAC-Bayesian binary activated deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 6872–6882. Curran Associates, Inc., 2019.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *arXiv preprint arXiv:1906.10652*, 2019.

Matthias Seeger, John Langford, and Nimrod Megiddo. An improved predictive accuracy bound for averaging classifiers. In *Proceedings of the 18th International Conference on Machine Learning*, number CONF, pages 290–297, 2001.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. Non-vacuous generalization bounds at the ImageNet scale: A PAC-Bayesian compression approach. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJgqqsAct7.

A Further Experimental Details

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:

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

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

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

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

A.2 Reinforce Model

During evaluation, the **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 the baseline **reinforce** suggest themselves, arising from different approximations to the Q-expected loss of the minibatch, $B\subseteq S$ (with data indices $\mathscr{B}$). From the identities

$$\nabla_\phi\mathbb{E}_{\theta\sim q_\phi}R_B(f_\theta)=\mathbb{E}_{\theta\sim q_\phi}\frac{1}{|B|}\sum_{i\in\mathscr{B}}\ell(f_\theta(\mathbf{x}_i),y_i)\nabla_\phi\log q_\phi(\theta)=\frac{1}{|B|}\sum_{i\in\mathscr{B}}\mathbb{E}_{\theta\sim q_\phi}\ell(f_\theta(\mathbf{x}_i),y_i)\nabla_\phi\log q_\phi(\theta)$$

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

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

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 Kingma et al., 2015, for more information).

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 (7). 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.

A.3 Dataset Details

We used the MNIST dataset version 3.0.1, available online at `http://yann.lecun.com/exdb/mnist/`, which contains 60000 training examples and 10000 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 $[0,1]$.

Table 2: 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.4. Run times are rounded to nearest 5 minutes.

	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	15020	13613	2363	3571	3011	5561	3204	4000
Runtime/min	10	5	40	40	35	30	25	35	30	25

A.4 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 the original paper, 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`.

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 $\in \{1, 2, 3\}$.

A.5 Final Hyperparameter Settings

In Table 2 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.

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, $\lambda = m = 60000$.
- Number of Hidden Layers $= 3$ for all models, except PBGNet $\in \{1, 2, 3\}$.

A.6 Implementation and Runtime

Experiments were implemented using Python and the TensorFlow library (Abadi et al., 2015). Reported approximate runtimes are for execution on a NVIDIA GeForce RTX 2080 Ti GPU.