Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

Tighter risk certificates for neural networks

María Pérez-Ortiz MARIA.PEREZ@UCL.AC.UK
AI Centre, University College London (UK)

Omar Rivasplata RIVASPLATA@GOOGLE.COM
DeepMind (UK)

John Shawe-Taylor J.SHAWE-TAYLOR@UCL.AC.UK
AI Centre, University College London (UK)

Csaba Szepesvári SZEPI@GOOGLE.COM
DeepMind (UK)

Editor: Kevin Murphy and Bernhard Schölkopf

Abstract

This paper presents an empirical study regarding training probabilistic neural networks using training objectives derived from PAC-Bayes bounds. In the context of probabilistic neural networks, the output of training is a probability distribution over network weights. We present two training objectives, used here for the first time in connection with training neural networks. These two training objectives are derived from tight PAC-Bayes bounds. We also re-implement a previously used training objective based on a classical PAC-Bayes bound, to compare the properties of the predictors learned using the different training objectives. We compute risk certificates that are valid on any unseen examples for the learnt predictors. We further experiment with different types of priors on the weights (both data-free and data-dependent priors) and neural network architectures. Our experiments on MNIST and CIFAR-10 show that our training methods produce competitive test set errors and non-vacuous risk bounds with much tighter values than previous results in the literature, showing promise not only to guide the learning algorithm through bounding the risk but also for model selection. These observations suggest that the methods studied here might be good candidates for self-certified learning, in the sense of certifying the risk on any unseen data without the need for data-splitting protocols.

Keywords: Deep learning, neural work training, weight randomization, generalization, pathwise reparametrized gradients, PAC-Bayes with Backprop, data-dependent priors.

1. Introduction

In a probabilistic neural network, the result of the training process is a distribution over network weights, rather than simply fixed weights. Several prediction schemes can be devised based on a probability distribution over weights. For instance, one may use a randomized predictor, where each prediction is done by randomly sampling the weights from the data-dependent distribution obtained as the result of the training process. Another possible scheme consists of predicting with the mean of the learned distribution. Yet another prediction scheme is based on integrating the predictions of all possible parameter settings, weighted according to the learned distribution.

In this paper we experiment with probabilistic neural networks from a PAC-Bayesian approach. We name 'PAC-Bayes with Backprop' (PBB) the family of (probabilistic) neural network training methods derived from PAC-Bayes bounds and optimized through stochastic gradient descent. The work reported here is the result of our empirical studies undertaken to investigate three PBB training objectives. For reference, they are the functions f_{quad}, f_{lambda} and f_{classic}, shown respectively in Eq. (9), Eq. (10) and Eq. (11) below. These objectives are based on PAC-Bayes bounds with similar names, which are relaxations of the PAC-Bayes relative entropy bound (Langford and Seeger, 2001), also known as the PAC-Bayes-kl bound in the literature. The classic PAC-Bayes bound, from which f_{classic} is derived, is that of McAllester (1999), but we use the improved dependence on the number of training patterns as suggested by Maurer (2004). The PAC-Bayes-lambda bound is that of Thiemann et al. (2017). The PAC-Bayes-quadratic bound, from which f_{quad} is derived, was first introduced by us in the preprint Rivasplata et al. (2019b). Our final aim is to provide tight risk certificates on the classification error of the randomized classifiers generated by these training methods. The certificate is valid on unseen examples, making it a first step towards self-certified learning, which would avoid the need for data splitting protocols in both testing and model selection.

Our line of research owes credit to previous works that have trained a probabilistic neural network by minimizing a PAC-Bayes bound, or used a PAC-Bayes bound to give risk certificates for trained neural networks. Langford and Caruana (2001) developed a method to train a probabilistic neural network by randomizing the weights with Gaussian noise (adjusted in a data-dependent way via a sensitivity analysis), and computed an upper bound on the error using the PAC-Bayes-kl bound.[1] They also pointed that PAC-Bayes bounds might be fruitful for computing non-vacuous generalization bounds for neural nets. Dziugaite and Roy (2017) used a training objective (essentially equivalent to f_{classic}) based on a relaxation of the PAC-Bayes-kl bound. They optimized this objective using stochastic gradient descent (SGD), and computed a confidence bound on the error of the randomized classifier following the same approach that Langford and Caruana (2001) used to compute their error bound. Dziugaite and Roy (2018) developed a two-stage method to train a probabilistic neural net, which in the first stage trains a prior mean by empirical risk minimization via stochastic gradient Langevin dynamics (Welling and Teh, 2011), and in the second stage re-uses the same training data used for the prior in order to train a posterior Gaussian distribution over weights by minimizing a relaxation of the classic PAC-Bayes bound which accounts for the data re-use.

In this paper we report experiments on MINIST and CIFAR-10 with the three training objectives mentioned above. We used by default the randomized predictor scheme (also called the 'stochastic predictor' in the PAC-Bayes literature), justified by the fact that PAC-Bayes bounds give high-confidence guarantees on the expected loss of the randomized predictor. Since training is based on a surrogate loss function, optimizing a PBB objective gives a high-confidence guarantee on the randomized predictor's risk under the surrogate

[1] Inversion of the PAC-Bayes-kl bound (we explain this in Section 6) gives a certificate (upper bound) on the risk of the randomized predictor, in terms of its empirical error and other quantities. The empirical error term is evaluated indirectly by Monte Carlo sampling, and a bound on the tail of the Monte Carlo evaluation (Langford and Caruana, 2001, Theorem 2.5) is combined with the PAC-Bayes-kl bound to give a numerical risk certificate that holds with high probability over data and Monte Carlo samples.

loss. Accordingly, to obtain guarantees that are valid for the classification (zero-one) loss, we separately evaluate the test set error and compute a confidence bound on the risk based on this loss (following the same procedure that was used by Langford and Caruana (2001)). For the sake of comparison we also report the test set evaluations of the other two predictor schemes described above, namely, the mean and the ensemble predictors.

Our work took inspiration from Blundell et al. (2015), whose results showed that randomized weights achieve competitive test set errors; and from Dziugaite and Roy (2017, 2018), whose results gave randomized neural network classifiers with reasonable test set errors and, more importantly, non-vacuous risk bound values. Our experiments show that PBB training objectives can (a) achieve competitive test set errors (e.g. comparable to Blundell et al. (2015) and empirical risk minimisation), while also (b) deliver risk certificates with reasonably tight values. Our results show as well a significant improvement over those of Dziugaite and Roy (2017, 2018): we further close the gap between the risk certificate (bound value) and the risk estimate (test set error rate). As we argue below, this improvement comes from the tightness of the PAC-Bayes bounds we used, which is established analytically and corroborated by our experiments on MNIST and CIFAR-10 with deep fully connected networks and convolutional neural networks.

Regarding the tightness of the training objectives, Dziugaite and Roy's training objective (which in our notation takes essentially the form of f_{classic} shown in Eq. (11) below) has the disadvantage of being sub-optimal in the regime of small losses. This is because to obtain it they relaxed the PAC-Bayes-kl bound via an inequality that is loose in this regime. The looseness was the price paid for having a computable objective. Note that small losses is precisely the regime of interest in neural network training (although the true loss being small is dataset and architecture dependent). By contrast, our proposed training objectives (f_{quad} and f_{lambda} in Eq. (9) and Eq. (10) below) are based on relaxing the PAC-Bayes-kl bound by an inequality that is tighter in this same regime of small losses, which is one of the reasons explaining our tighter risk certificates in MNIST (not for CIFAR-10, which could be explained by the large empirical loss obtained at the end of the optimisation). Interestingly, our own re-implementation of f_{classic} also gave improved results compared to the results of Dziugaite and Roy, which suggests that besides the training objectives we used, also the training strategies we used are responsible for the improvements.

A clear advantage of PAC-Bayes with Backprop (PBB) methods is being an instance of self-certified[2] learning: When training probabilistic neural nets by PBB methods the output is not just a predictor but simultaneously a *tight risk certificate* that guarantees the quality of predictions on unseen examples. The value of self-certified learning algorithms is in the possibility of using of all the available data to achieve both goals (learning a predictor and certifying its risk) simultaneously, thus obviating the use of data-splitting protocols. Note that risk certificates *per se* will not impress until their reported values match or closely follow the classification error rates evaluated on a test set, so that the risk certificate is informative of the error on unseen examples. This is where our work makes a significant contribution, since our PBB training methods lead to much tighter risk certificates for neural nets than previous works in the literature. Once again, the data-dependent solution found by our optimization process comes together with a high-confidence guarantee that

[2] A learning method is self-certified if it uses all the available data in order to output a hypothesis and simultaneously a tight certificate on its risk that is valid in unseen examples (cf. Freund (1998)).

certifies its risk under the surrogate training loss, and to obtain a high-confidence guarantee under the classification (zero-one) loss we evaluate *post training* a confidence bound on the classification error. A more ambitious goal would be to establish calibration[3] of the surrogate cross-entropy loss, so that its minimization guarantees minimal classification loss.

We would like to highlight the elegant simplicity of the methods presented here: Our results are achieved i) with priors learnt through empirical risk minimisation of the surrogate loss on a subset of the dataset (which does not overlap with the data used for computing the risk certificate for the probabilistic neural network, thus in line with classical PAC-Bayes priors) and ii) via classical SGD optimization. In contrast, Dziugaite and Roy (2018) trained a special type of data-dependent PAC-Bayes prior on the whole dataset using SGLD optimization. They justified this procedure arguing that the limit distribution of SGLD satisfies the differential privacy property (but a finite-time guarantee was missing), and relaxed the PAC-Bayes bound with a correction term based on the concept of max-information[4] to account for using the same data to train the prior mean and to evaluate the PAC-Bayes bound. Furthermore, our methods do not involve tampering with the training objective, as opposed to Blundell et al. (2015), who used a “KL attenuating trick” by inserting a tunable parameter as a factor of the Kullback-Leibler (KL) divergence in their objective. Our work highlights the point that it is worthwhile studying simple methods, not just to understand their scope or for the sake of having a more controlled experimental setup, but also to more accurately assess the real value added by the ‘extras’ of the more complex methods.

Our contributions:

1. We rigorously study and illustrate ‘PAC-Bayes with Backprop’ (PBB), a generic strategy to derive neural network training methods from PAC-Bayes bounds.

2. We propose –and experiment with– two new PBB training objectives: one derived from the PAC-Bayes-quadratic bound of Rivasplata et al. (2019b), and one derived from the PAC-Bayes-lambda bound of Thiemann et al. (2017).

3. We also re-implement the training objective used by Dziugaite and Roy for the sake of comparing our training objectives and training strategy, both with respect to test set accuracy and risk certificates obtained.

4. We connect PAC-Bayes with Backprop (PBB) methods to the Bayes-by-Backprop (BBB) method of Blundell et al. (2015), inspired by Bayesian learning, which achieved competitive test set accuracy. Unlike BBB, our training methods not only achieve competitive test set errors, but require less heuristics and also provide a risk certificate.

5. We demonstrate via experimental results that PBB methods might be able to achieve self-certified learning with nontrivial certificates: obtaining competitive test set errors and computing non-vacuous bounds with much tighter values than previous works.

Broader context. Deep learning is a vibrant research area. The success of deep neural network models in several tasks has motivated many works that study their optimization and generalization properties (some of the collective knowledge is condensed in a few recent sources such as Montavon et al. (2012); Goodfellow et al. (2016); Aggarwal (2018)).

[3] Akin to results on calibration of the surrogate hinge loss, cf. Steinwart and Christmann (2008).

[4] Dwork et al. (2015a,b) proposed this concept in the context of adaptive data analysis.

Some works focus on experimenting with methods to train neural networks, others aim at generating knowledge and understanding about these fascinating learning systems. In this paper we intend to contribute both ways. We focus on supervised classification problems through probabilistic neural networks, and we experiment with training objectives that are principled and consist of interpretable quantities. Furthermore, our work puts an emphasis on certifying the quality of predictions beyond a specific data set.

Note that known neural network training methods range from those that have been developed based mainly on heuristics to those derived from sound principles. Bayesian learning, for instance, offers principled approaches for learning data-dependent distributions over network weights (see e.g. Buntine and Weigend (1991), Neal (1992), MacKay (1992), Barber and Bishop (1998)), hence probabilistic neural nets arise naturally in this approach. Bayesian neural networks continue to be developed, with notable recent contributions e.g. by Hernández-Lobato and Adams (2015); Martens and Grosse (2015); Blundell et al. (2015); Gal and Ghahramani (2016); Louizos and Welling (2016); Ritter et al. (2018), among others. Our work is complementary of Bayesian learning in the sense that our methods also offer principled training objectives for learning probabilistic neural networks. However, there are differences between the PAC-Bayesian and Bayesian approaches that are important to keep in mind (we discuss the differences in Section 3). It is worth mentioning also that some works have pointed out the resemblance between the Bayesian evidence lower bound (ELBO) and PAC-Bayes bounds, and made connections between the PAC-Bayes approach and variational Bayesian inference (Achille and Soatto, 2018; Thakur et al., 2019).

As we pointed out before, we are not the first to train a probabilistic neural network by minimizing a PAC-Bayes bound, or to use a PAC-Bayes bound to give risk certificates for trained neural networks. We already mentioned Langford and Caruana (2001) and Dziugaite and Roy (2017, 2018), whose works have directly influenced ours. Next, we comment on some other works that connect PAC-Bayes with neural networks. London (2017) approached the generalization of neural networks by a stability-based PAC-Bayes analysis, and proposed an adaptive sampling algorithm for SGD that optimizes its distribution over training instances using multiplicative weight updates. Neyshabur et al. (2017, 2018) examined the connection between some specifically defined complexity measures and generalization, the part related to our work is that they specialized a form of the classic PAC-Bayes bound and used Gaussian noise on network weights to give generalization bounds for probabilistic neural networks based on the norms of the weights. Zhou et al. (2019) compressed trained networks by pruning weights to a given target sparsity, and gave generalization guarantees on the compressed networks, which were based on randomizing predictors according to their 'description length' and a specialization of Catoni (2007)'s PAC-Bayes bound.

We would like to point out that the present work builds on Rivasplata et al. (2019b). In the meantime, more works have appeared that connect neural networks with PAC-Bayes bounds in various settings: Letarte et al. (2019), Viallard et al. (2019), Lan et al. (2020), Dziugaite et al. (2020), Biggs and Guedj (2020). We do not elaborate on these works as they deal with significantly different settings than ours (e.g. they study binary activated networks or ensembles or focus exclusively on the role of the prior).

Paper layout. The rest of the paper is organized as follows. In Section 2 we briefly recall some notions of supervised learning, mainly to set the notation used later. In Section 3 we outline the PAC-Bayes framework and discuss some PAC-Bayes bounds, while in Section 5 we present the training objectives derived from them. Section 4 discusses the connection between our work and Blundell et al. (2015). The technical Section 6 describes the binary KL inversion strategy and the ways we use it. In Section 7 we present our experimental results. We conclude and discuss future research directions in Section 8.

2. Generalization through risk upper bounds

An algorithm that trains a neural network receives a finite list of training examples and produces a data-dependent weight vector $\hat{w} \in \mathcal{W} \subset \mathbb{R}^p$, which is used to make predictions on unseen examples. The ultimate goal is for the algorithm to find a weight vector that generalizes well, meaning that the decisions arrived at by using the learned $\hat{w}$ should give rise to a small loss on unseen examples.[5] Turning this into precise statements requires a formal description of the learning setting, briefly discussed next. The reader familiar with learning theory can skip the next couple of paragraphs and come back if they need clarifications regarding notation.

The training algorithm receives a size-n random sample $S = (Z_1, \dots, Z_n)$. Each example Z_i is randomly drawn from a space $\mathcal{Z}$ according to an underlying (but unknown) probability distribution[6] $P \in \mathcal{M}_1(\mathcal{Z})$. The example space usually takes the form $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ in supervised learning, where $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}$, each example being a pair $Z_i = (X_i, Y_i)$ consisting of an input X_i and its corresponding label Y_i. A space $\mathcal{W} \subseteq \mathbb{R}^p$ encompasses all possible weights, and it is understood that each possible weight vector $w \in \mathcal{W}$ maps to a predictor function $h_w : \mathcal{X} \to \mathcal{Y}$ that will assign a label $h_w(X) \in \mathcal{Y}$ to each new input $X \in \mathcal{X}$. While statistical inference is largely concerned with elucidating properties of the unknown data-generating distribution, the main focus of machine learning is on the quality of predictions, measured by the expected loss on unseen examples, also called the risk:

$$L(w) = \mathbb{E}[\ell(w, Z)] = \int_{\mathcal{Z}} \ell(w, z) P(dz)\,. \tag{1}$$

Here $\ell : \mathcal{W} \times \mathcal{Z} \to [0, \infty)$ is a fixed loss function. With these components, regression is defined as the problem when $\mathcal{Y} = \mathbb{R}$ and the loss function is the squared loss, namely $\ell(w, z) = (y - h_w(x))^2$ where $z = (x, y)$ is the input-label pair, while binary classification is the problem where $\mathcal{Y} = \{0, 1\}$ (or $\mathcal{Y} = \{-1, +1\}$) and the loss is set to be the zero-one loss: $\ell(w, z) = \mathbb{I}[y \neq h_w(x)]$.

The goal of learning is to find a weight vector with small risk $L(w)$. Since the data-generating distribution P is unknown, $L(w)$ is an unobservable objective. Replacing the expected loss with the average loss on the data gives rise to an observable objective called the *empirical risk* functional:

$$\hat{L}_S(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, Z_i)\,. \tag{2}$$

[5] In statistical learning theory the meaning of *generalization* of a learning method has a precise definition (see e.g. Shalev-Shwartz and Ben-David (2014)). We use the word in a slightly broader sense here.

[6] $\mathcal{M}_1(\mathcal{Z})$ denotes the set of all probability measures over $\mathcal{Z}$.

In practice, the minimization of $\hat{L}_S$ is often done with some version of gradient descent. Since the zero-one loss gives rise to a piecewise constant loss function, which is provably hard to optimize, in classification it is common to replace it with a smooth(er) loss, such as the cross-entropy loss, while changing the range of h_w to $[0,1]$.

Under certain conditions, a small empirical risk leads to a weight that is guaranteed to have a small risk gap[7]. Examples of such conditions are when the set of functions $\{h_w : w \in \mathbb{R}^p\}$ representable has a small capacity relative to the sample size, or the map that produces the weights given the data is stable. However, often minimizing the empirical risk can lead to a situation where the risk of the learned weight is significantly larger than the empirical risk – a case of overfitting. To prevent overfitting, various methods are commonly used. These include complexity regularization, early stopping, injecting noise in various places into the learning process, etc (e.g. Srivastava et al. (2014), Wan et al. (2013), Caruana et al. (2001), Hinton and van Camp (1993)).

An alternative to these is to minimize a surrogate objective which is guaranteed to give an upper bound on the risk. As long as the upper bound is tight and the optimization gives rise to a small value for the surrogate objective, the user can be sure that the risk will also be small: In this sense, overfitting is automatically prevented, while we also automatically get a self-certified learning method (cf. Freund (1998), Langford and Blum (2003)). In this paper we follow this last approach, with two specific training objectives derived from corresponding PAC-Bayes bounds, which we introduce in the next section. The approach to learning data-dependent distributions over hypotheses by minimizing a PAC-Bayes bound is mentioned already in McAllester (1999), credit for this approach in various contexts is due also to Germain et al. (2009), Seldin and Tishby (2010), Keshet et al. (2011), Noy and Crammer (2014), Keshet et al. (2017), among others. Subsequent use of this approach for training neural nets was done by Dziugaite and Roy (2017, 2018).

As will be demonstrated below, our experiments based on our two training objectives f_{quad} and f_{lambda} (Eq. (9) and Eq. (10) below) lead to (a) test set performance comparable to that of Blundell et al. (2015), while (b) computing non-vacuous bounds with tighter values than those obtained by f_{classic} (Eq. (11) below) which is essentially equivalent to the training objective used by Dziugaite and Roy.

3. PAC-Bayes bounds

Probabilistic neural networks are realized as probability distributions over the weight space. While the outcome of a classical (non-probabilistic) neural network training method is a data-dependent weight vector, the outcome of training a probabilistic neural network is a data-dependent distribution[8] over weights, say Q_S. Then, given a fresh input X, the network predicts its label by drawing a weight vector W at random according to Q_S and applying the predictor h_W to X. Each new prediction requires a fresh draw. One way, which we adopt in this paper, to measure the performance of the resulting randomizing predictor, is to use the expected loss over the random draws of weights. Accordingly, the average empirical loss becomes $Q_S[\hat{L}_S] = \int_{\mathcal{W}} \hat{L}_S(w) Q_S(dw)$ and the average population loss

[7] The risk gap is the difference between the risk (1) and the empirical risk (2).

[8] Formally, a data-dependent distribution over $\mathcal{W}$ is a stochastic kernel from $\mathcal{S}$ to $\mathcal{W}$. This formalization of data-dependent distributions over predictors is covered lucidly by Rivasplata et al. (2019a).

becomes $Q_S[L] = \int_{\mathcal{W}} L(w) Q_S(dw)$. In general, we denote by $\rho[f]$ the integral $\int_{\mathcal{W}} f(w)\rho(dw)$ whenever ρ is a probability distribution over $\mathcal{W}$ and $f : \mathcal{W} \to \mathbb{R}$ an integrable function.

To introduce the promised PAC-Bayes bounds we need to recall some further definitions. Given two probability distributions $Q, Q' \in \mathcal{M}_1(\mathcal{W})$, the Kullback-Leibler (KL) divergence of Q from Q', also known as relative entropy of Q given Q', is defined as follows:

$$\mathrm{KL}(Q\|Q') = \int_{\mathcal{W}} \log\Big(\frac{dQ}{dQ'}\Big)\, dQ$$

when dQ/dQ', the Radon-Nikodym derivative of Q with respect to Q', is defined; otherwise $\mathrm{KL}(Q\|Q') = \infty$. For $q, q' \in [0, 1]$ we will write

$$\mathrm{kl}(q\|q') = q \log(\frac{q}{q'}) + (1-q)\log(\frac{1-q}{1-q'}) \tag{3}$$

which is called the binary KL divergence, and is the divergence of the Bernoulli distribution with parameter q from the Bernoulli distribution with parameter q'.

The PAC-Bayes-kl theorem (Langford and Seeger (2001), Seeger (2002), Maurer (2004)) concludes that as long as the loss takes values in $[0, 1]$, for any $\delta \in [0, 1]$, with probability at least $1 - \delta$ over random samples S, for any distribution Q over $\mathcal{W}$ it holds that

$$\mathrm{kl}(Q[\hat{L}_S]\|Q[L]) \leq \frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n}\,, \tag{4}$$

where Q^0 is a data-free distribution over $\mathcal{W}$, which means that Q^0 is fixed without any dependence on the data on which the bound is evaluated. One can lower-bound the binary KL divergence, e.g., using the relaxed version of Pinsker's inequality $\mathrm{kl}(\hat{p}\|p) \geq 2(p - \hat{p})^2$, and then solve the resulting inequality for $Q[L]$ (see e.g. Tolstikhin and Seldin (2013)). Alternatively, one may use the refined version of Pinsker's inequality $\mathrm{kl}(\hat{p}\|p) \geq (p-\hat{p})^2/(2p)$ valid for $\hat{p} < p$ (see e.g. Boucheron et al. (2013, Lemma 8.4)), which is tighter when $p < 1/4$ (see Section 5.1), and thus get

$$Q[L] - Q[\hat{L}_S] \leq \sqrt{2Q[L]\frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n}}\,. \tag{$\star$}$$

The difference to the result one gets from the relaxed version of Pinsker's inequality is the appearance of $Q[L]$ under the square root. This, in particular, tells us that the inequality is tighter when the population loss, $Q[L]$, is smaller (specifically when $Q[L] < 1/4$). But it is exactly because of the appearance of $Q[L]$ on the right-hand side that this bound is not immediately useful for optimization purposes. However, one can view the above inequality as a quadratic inequality on $\sqrt{Q[L]}$. Solving this inequality for $Q[L]$ leads to the following empirical PAC-Bayes bound, which to the best of our knowledge is new:

Theorem 1 *For any n, for any $P \in \mathcal{M}_1(\mathcal{X})$, for any data-free distribution $Q^0 \in \mathcal{M}_1(\mathcal{W})$, for any loss function with range $[0, 1]$, for any $\delta \in (0, 1)$, with probability $\geq 1 - \delta$ over size-n i.i.d. samples $S \sim P_1^n$, simultaneously for all distributions Q over $\mathcal{W}$ we have*

$$Q[L] \leq \left(\sqrt{Q[\hat{L}_S] + \frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} + \sqrt{\frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}\right)^2\,. \tag{5}$$

Alternatively, using ($\star$) combined with the inequality $\sqrt{ab} \leq \frac{1}{2}(\lambda a + \frac{b}{\lambda})$ valid for all $\lambda > 0$, after some derivations one obtains the PAC-Bayes-λ bound of Thiemann et al. (2017):

Theorem 2 *For any n, for any $P \in \mathcal{M}_1(\mathcal{X})$, for any data-free distribution $Q^0 \in \mathcal{M}_1(\mathcal{W})$, for any loss function with range $[0,1]$, for any $\delta \in (0,1)$, with probability $\geq 1-\delta$ over size-n i.i.d. samples $S \sim P_1^n$, simultaneously for all distributions Q over $\mathcal{W}$ and $\lambda \in (0,2)$ we have*

$$Q[L] \leq \frac{Q[\hat{L}_S]}{1-\lambda/2} + \frac{\mathrm{KL}(Q\|Q^0) + \log(2\sqrt{n}/\delta)}{n\lambda(1-\lambda/2)}\,. \tag{6}$$

For convenience, we quote the classical PAC-Bayes bound of McAllester (1999):

Theorem 3 *For any n, for any $P_1 \in \mathcal{M}_1(\mathcal{Z})$, for any data-free distribution $Q^0 \in \mathcal{M}_1(\mathcal{H})$, for any loss function with range $[0,1]$, for any $\delta \in (0,1)$, with probability $\geq 1-\delta$ over size-n i.i.d. samples $S \sim P_1^n$, simultaneously for all distributions Q over $\mathcal{W}$ we have*

$$Q[L] \leq Q[\hat{L}_S] + \sqrt{\frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}\,. \tag{7}$$

The original proof of McAllester (1999) gave a slightly looser bound. The form presented in Theorem 3 is with the sharp dependence on n due to Maurer (2004).

Notice that the conclusion of these theorems is an upper bound on $Q[L]$ that holds simultaneously for all distributions Q over weights, with high probability (over samples). In particular, the bounds allow to choose a distribution Q_S in a data-dependent manner, which is why they are usually called 'posterior' distributions in the PAC-Bayesian literature. These theorems could be re-written directly in terms of the data-dependent distributions Q_S represented as stochastic kernels (as done by Rivasplata et al. (2019a)). Below in Section 5 we discuss training objectives derived from these bounds. Notice that there are many other PAC-Bayes bounds available in the literature (the usual ones are by McAllester (1999), Langford and Seeger (2001), Catoni (2007); but see also McAllester (2003), Keshet et al. (2011), the mini tutorial of van Erven (2014) and the primer of Guedj (2019)). Each such bound readily leads to a training objective by replicating the procedure described below.

4. The Bayes by Backprop (BBB) objective

The 'Bayes by backprop' (BBB) method of Blundell et al. (2015) is inspired by a variational Bayes argument (Fox and Roberts, 2012), where the idea is to learn a distribution over the weights that approximate the Bayesian posterior distribution. Choosing a p-dimensional Gaussian $Q_\theta = \mu + \sigma\mathcal{N}(0,I)$, parametrized by $\theta = (\mu, \sigma)$, the optimum parameters are those that minimize $\mathrm{KL}(Q_\theta \| P(\cdot|S))$, i.e. the KL divergence from Q_θ and the Bayesian posterior $P(\cdot|S)$. By a simple calculation, and using the Bayes rule, one can extract:

$$\mathrm{KL}(Q_\theta\|P(\cdot|S)) = \int_{\mathcal{W}} -\log(P(S|w))Q_\theta(dw) + \mathrm{KL}(Q_\theta\|Q^0),$$

where Q^0 stands here for the Bayesian prior distribution. Thus, minimizing $\mathrm{KL}(Q_\theta\|P(\cdot|S))$ is equivalent to minimizing the right-hand side, which presents a sum of a data-dependent term (the expected negative log-likelihood) and a prior-dependent term ($\mathrm{KL}(Q_\theta \| Q^0)$), which makes this optimization problem analogous to that of minimizing a PAC-Bayes bound, since the latter also balances a fit-to-data (empirical loss) term and a fit-to-prior (KL) term. However, in the PAC-Bayes framework the 'prior' is a reference distribution and the 'posterior' does not need to be derived from the prior by an update factor, but is rather unrestricted. This is a crucial difference with Bayesian learning, and one that makes the PAC-Bayes framework a lot more flexible in the choice of distributions, even compared to generalized Bayesian approaches (Bissiri et al., 2016).

As we mentioned before, the training objective proposed by Blundell et al. (2015) is inspired by the variational Bayesian argument outlined above, in particular, in our notation the training objective they proposed and experimented with is as follows:

$$f_{\mathrm{bbb}}(Q) = Q[\hat{L}_S] + \eta \, \frac{\mathrm{KL}(Q \| Q^0)}{n} \,. \tag{8}$$

The scaling factor, $\eta > 0$, is introduced in a heuristic manner to make the method more flexible, while the variational Bayes argument gives (8) with $\eta = 1$. When η is treated as a tuning parameter, the method can be interpreted as searching in "KL balls" centered at Q^0 of various radii. Thus, the KL term then plays the role of penalizing the complexity of the model space searched. Blundell et al. (2015) propose to optimize this objective (for a fixed η) using stochastic gradient descent (SGD), which randomizes over both mini-batches and the weights, and uses the pathwise gradient estimate (Price, 1958). The resulting gradient-calculation procedure can be seen to be only at most twice as expensive as standard backpropagation – hence the name of their method. The hyperparameter $\eta > 0$ is chosen using a validation set, which is also often used to select the best performing model among those that were produced during the course of running SGD (as opposed to using the model obtained when the optimization procedure finishes).

The results in Blundell et al. (2015) have shown that probabilistic neural networks enable an intuitive and principled implementation of classification reject options (i.e. allow the model say "I don't know" when the classification uncertainty for a new example is higher than a certain threshold) and model pruning. The use of a prior during training has empirically shown similar results to other implicit regularisation schemes, such as dropout. Finally, their weight uncertainty was also used to drive the exploration-exploitation trade-off in reinforcement learning.

5. Towards practical PAC-Bayes with Backprop (PBB) methods

The essential idea of "PAC-Bayes with Backprop" (PBB) is to train a probabilistic neural network by minimizing an upper bound on the risk, specifically, a PAC-Bayes bound. Here we present two training objectives, derived from Eq. (5) and Eq. (6) respectively, in the context of *classification problems* when the loss is the zero-one loss or a surrogate loss. These objectives are used here for the first time to train probabilistic neural networks. We also discuss the training objective derived from Eq. (7) for comparison purposes.

To optimize the weights of neural networks the standard idea is to use a form of stochastic gradient descent, which requires the ability to efficiently calculate gradients of the objective to be optimized. When the loss is the zero-one loss, $w \mapsto \hat{L}^{01}_S(w)$, the training loss viewed as a function of the weights, is piecewise constant, which makes simple gradient-based methods fail (since the gradient, whenever it exists, is zero). As such, it is customary to replace the zero-one loss with a smoother "surrogate loss" that plays well with gradient-based optimization. In particular, the standard loss used on multiclass classification problems is the cross-entropy loss, $\ell^{\text{x-e}} : \mathbb{R}^k \times [k] \to \mathbb{R}$ defined by $\ell^{\text{x-e}}(z, y) = -\log(\sigma(z)_y)$ where $z \in \mathbb{R}^k$, $y \in [k] = \{1, \dots, k\}$ and $\sigma : \mathbb{R}^k \to [0, 1]^k$ is the soft-max function defined by $\sigma(z)_i = \exp(z_i)/\sum_j \exp(z_j)$. This choice can be justified on the grounds that $\ell^{\text{x-e}}(z, y)$ gives an upper bound on the probability of mistake when the label is chosen at random from the distribution produced by applying soft-max on z (e.g., the output of the last linear layer of a neural network).[9] We thus also propose to replace the zero-one loss with the cross-entropy loss in either Theorem 1 or Theorem 2, leading to the objectives

$$f_{\text{quad}}(Q) = \left(\sqrt{Q[\hat{L}_S^{\text{x-e}}] + \frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} + \sqrt{\frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \right)^2 \tag{9}$$

and

$$f_{\text{lambda}}(Q, \lambda) = \frac{Q[\hat{L}_S^{\text{x-e}}]}{1 - \lambda/2} + \frac{\text{KL}(Q\|Q^0) + \log(2\sqrt{n}/\delta)}{n\lambda(1 - \lambda/2)} \tag{10}$$

where $\hat{L}_S^{\text{x-e}}(w) = \frac{1}{n}\sum_{i=1}^n \tilde{\ell}_1^{\text{x-e}}(h_w(X_i), Y_i)$ denotes the empirical error rate under the 'bounded' version of cross-entropy loss, namely the loss $\tilde{\ell}_1^{\text{x-e}}$ described below, and $h_w : \mathcal{X} \to \mathbb{R}^k$ denotes the function implemented by the neural network that uses weights w.

For comparison, the training objective derived from Theorem 3 takes the following form:

$$f_{\text{classic}}(Q) = Q[\hat{L}_S^{\text{x-e}}] + \sqrt{\frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}\,. \tag{11}$$

The next issue to address is that the cross-entropy loss is unbounded, while the previous theorems required a bounded loss. This is fixed by enforcing an upper bound on the cross-entropy loss by lower-bounding the network probabilities by a value $p_{\min} > 0$ (Dziugaite and Roy, 2018). This is achieved by replacing σ in the definition of $\ell^{\text{x-e}}$ by $\tilde{\sigma}(z)_y = \max(\sigma(z)_y, p_{\min})$. This adjustment gives a 'bounded cross-entropy' loss function $\tilde{\ell}^{\text{x-e}}(z, y) = -\log(\tilde{\sigma}(z)_y)$ with range between 0 and $\log(1/p_{\min})$. Finally, re-scaling by $1/\log(1/p_{\min})$ gives a loss function $\tilde{\ell}_1^{\text{x-e}}$ with range [0,1] ready to be used in the PAC-Bayes bounds and training objectives discussed here. The latter ($\tilde{\ell}_1^{\text{x-e}}$) is used as the surrogate loss for training in all our experiments with f_{quad}, f_{lambda}, and f_{classic}.

Optimization of (9) entails minimizing over Q only, while optimization of (10) is done by alternating minimization with respect to Q and λ, similar to the procedure that was used by Thiemann et al. (2017) in their experiments with SVMs. By choosing Q appropriately, in either case we use the pathwise gradient estimator (Price, 1958; Jankowiak and Obermeyer, 2018) as done by Blundell et al. (2015). In particular, assuming that $Q = Q_\theta$ with $\theta \in \mathbb{R}^q$ is such that $h_W(\cdot)$ with $W \sim Q_\theta$ ($W \in \mathbb{R}^p$) has the same distribution as $h_{f_\theta(V)}(\cdot)$ where

[9] Indeed, owning to the inequality $\log(x) \leq x - 1$, which is valid for any $x > 0$, given any $z \in \mathbb{R}^k$ and $y \in [k]$, if $Y \sim \sigma(z)$ then $\mathbb{E}[\mathbb{I}\{Y \neq y\}] = \mathbb{P}(Y \neq y) = 1 - \sigma(z)_y \leq \ell^{\text{x-e}}(z, y)$.

Algorithm 1 PAC-Bayes with Backprop (PBB)

```
Require:
    $\mu_0$                                          ▷ Prior center parameters
    $\rho_0$                                         ▷ Prior scale hyper-parameter
    $Z_{1:n}$                                        ▷ Training examples (inputs + labels)
    $\delta \in (0,1)$                               ▷ Confidence parameter
    $\alpha \in (0,1)$, $T$                          ▷ Learning rate; Number of iterations
Ensure: Optimal $\mu, \rho$                          ▷ Centers, scales
procedure PB_QUAD_GAUSS
    $\mu \leftarrow \mu_0$                           ▷ Set init. posterior center to prior center
    $\rho \leftarrow \rho_0$                         ▷ Set init. posterior scale to prior scale
    for $t \leftarrow 1 : T$ do                      ▷ Run SGD for T iterations.
        Sample $V \sim \mathcal{N}(0, I)$
        $W = \mu + \log(1 + \exp(\rho)) \odot V$
        $f(\mu, \rho) = f_{\text{quad}}(Z_{1:n}, W, \mu, \rho, \mu_0, \rho_0, \delta)$
        SGD gradient step using $\begin{bmatrix} \nabla_\mu f \\ \nabla_\rho f \end{bmatrix}$
    end for
    return $\mu, \rho$
end procedure
```

$V \in \mathbb{R}^{p'}$ is drawn at random from a *fixed* distribution P_V and $f_\theta : \mathbb{R}^{p'} \to \mathbb{R}^p$ is a smooth map, an unbiased estimate of the gradient of the loss-map $\theta \mapsto Q_\theta[\ell(h_\bullet(x), y)]$ at some θ can be obtained by drawing $V \sim P_V$ and calculating $\frac{\partial}{\partial \theta} \ell(h_{f_\theta(V)}(x), y)$, thereby reducing the efficient computation of the gradient to the application of the backpropagation algorithm on the map $\theta \mapsto \ell(h_{f_\theta(v)}(x), y)$ at $v = V$.[10]

Following Blundell et al. (2015), the reparametrization we use is $W = \mu + \sigma \odot V$ with appropriate distribution (Gauss or Laplace) for each coordinate of V. The process of optimization is implemented using the transformation $\sigma = \log(1 + \exp(\rho))$ and the gradient updates are with respect to μ and ρ, as can be seen in Algorithm 1. Note that Algorithm 1 shows the procedure for optimising f_{quad} with Gaussian noise. The procedure with Laplace noise is similar. The procedure for f_{classic} is similar. The procedure for f_{lambda} would be very similar except that f_{lambda} has the additional parameter λ.

5.1 Pinsker inequality: relaxed versus refined

We explain now the differences between the relaxed and refined versions of the Pinsker inequality, used for defining the above presented PAC-Bayes inspired training objectives and crucial to understand their differences. We refer the reader to Eq. (3) for the definition of the binary KL divergence, denoted $\text{kl}(\cdot\|\cdot)$. The relaxed Pinsker inequality reads:

$$\text{kl}(\hat{p}\|p) \geq 2(p - \hat{p})^2 \quad \text{for } \hat{p}, p \in (0,1), \tag{12}$$

while the refined Pinsker inequality takes the form:

$$\text{kl}(\hat{p}\|p) \geq \frac{(p - \hat{p})^2}{2p} \quad \text{for } \hat{p}, p \in (0,1),\ \hat{p} < p. \tag{13}$$

[10] Indeed (e.g. Ruiz et al. (2016)), $\frac{\partial}{\partial \theta} \int Q_\theta(dw) \ell(h_w(x), y) = \frac{\partial}{\partial \theta} \int P_V(dv) \ell(h_{f_\theta(v)}(x), y) = \int P_V(dv) \frac{\partial}{\partial u} \ell(h_{f_\theta(v)}(x), y)$, where the interchange of the partial derivative and the integral is justified when the partial derivatives are integrable, which needs to be verified on a case-by-case basis.

One can compare these two inequalities, to find regime of $p, \hat{p}$ in which one is better than the other. The result of the comparison is that Eq. (12) (used in f_{classic}) is tighter whenever $p > 1/4$, and Eq. (13) (used in f_{quad}) is tighter whenever $p < 1/4$. They match if $p = 1/4$.

5.2 The choice of the prior distribution

We experiment both with priors centered at randomly initialised weights and priors learnt by empirical risk minimisation using the surrogate loss on a subset of the dataset which is independent of the subset used to compute the risk certificate. Note that all n training data are used by the learning algorithm (n_0 examples used to build the prior, n to learn the posterior and $n - n_0$ to evaluate the risk certificate). This is to avoid needing differentially private arguments to justify learning the prior (Dziugaite and Roy, 2018). Since the posterior is initialised to the prior, the learnt prior translates to the posterior being initialised to a large region centered at the empirical risk minimiser. Similar approaches for building data-dependent priors have been considered before in the PAC-Bayesian literature (Lever et al., 2013; Parrado-Hernández et al., 2012; Dziugaite and Roy, 2018).

5.2.1 Priors: Laplace versus Gauss

We describe here the two distributions considered for the network weights: Laplace and Gaussian. The Laplace density with mean parameter $\mu \in \mathbb{R}$ and with variance $b > 0$ is:

$$p(x) = (2b)^{-1} \exp\bigl(-\frac{|x-\mu|}{b}\bigr) .$$

The KL divergence for two Laplace distributions is

$$\mathrm{KL}(\mathrm{Lap}(\mu_1, b_1) \| \, \mathrm{Lap}(\mu_0, b_0)) = \log(\frac{b_0}{b_1}) + \frac{|\mu_1 - \mu_0|}{b_0} + \frac{b_1}{b_0} e^{-|\mu_1 - \mu_0|/b_1} - 1 . \tag{14}$$

For comparison, recall that the Gaussian density with mean parameter $\mu \in \mathbb{R}$ and variance $b > 0$ has the following form:

$$p(x) = (2\pi b)^{-1/2} \exp\bigl(-\frac{(x-\mu)^2}{2b}\bigr) .$$

The KL divergence for two Gaussian distributions is

$$\mathrm{KL}(\mathrm{Gauss}(\mu_1, b_1) \| \, \mathrm{Gauss}(\mu_0, b_0)) = \frac{1}{2}\Bigl(\log(\frac{b_0}{b_1}) + \frac{(\mu_1 - \mu_0)^2}{b_0} + \frac{b_1}{b_0} - 1\Bigr) . \tag{15}$$

The formulas (14) and (15) above are for the KL divergence between one-dimensional Laplace or Gaussian distributions. It is straightforward to extend them to, say, d-dimensional product distributions, corresponding to random vectors with independent components, as in this case the KL is equal to the sum of the KL divergences of the components.

6. Computing risk certificates

After optimising the distribution over network weights through the previously presented training objectives, we compute a risk certificate on the error of the stochastic predictor, following the procedure of Langford and Caruana (2001). This uses the PAC-Bayes-kl theorem. First we describe how to invert the binary KL from Eq. (4). For $x \in [0,1]$ and $b \in [0,\infty)$, the "inverse" of the binary entropy with respect to the second argument is:

$$f^\star(x,b) = \sup\{y \in [x,1] \,:\, \mathrm{kl}(x\|y) \leq b\}$$

This is easily seen to be well-defined. Furthermore, the crucial property that we rely on is that $\mathrm{kl}(x\|y) \leq b$ holds precisely when $y \leq f^\star(x,b)$.

Note that the function $f^\star$ provides a way for computing an upper bound on $Q[L]$ based on the PAC-Bayes-kl bound: For any $\delta \in (0,1)$, with probability at least $1-\delta$ over size-n random samples S we have:

$$Q[L] \leq f^\star\Big(Q[\hat{L}_S], \frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n}\Big)\,.$$

At this point, as noted by Langford and Caruana (2001), the difficulty is evaluating $Q[\hat{L}_S]$. This quantity is not computable. Since $f^\star$ is a monotonically increasing function of its first argument (when fixing the second argument), it suffices to upper-bound $Q[\hat{L}_S]$.

6.1 Estimating the empirical loss via Monte Carlo sampling

In fact, $f^\star$ is also used to estimate the empirical term $Q[\hat{L}_S]$ by random weight sampling: If $W_1,\ldots,W_m \sim Q$ are i.i.d. and $\hat{Q}_m = \sum_{j=1}^m \delta_{W_j}$ is the empirical distribution, then for any $\delta' \in (0,1)$, with probability at least $1-\delta'$ we have $\mathrm{kl}(\hat{Q}_m[\hat{L}_S]\|Q[\hat{L}_S]) \leq m^{-1}\log(2/\delta')$ (see Langford and Caruana (2001, Theorem 2.5)), hence by the inversion formula:

$$Q[\hat{L}_S] \leq f^\star\Big(\hat{Q}_m[\hat{L}_S], \frac{1}{m}\log(\frac{2}{\delta'})\Big)\,.$$

This expression can be applied to upper-bound $Q[\hat{L}_S^{01}]$ or $Q[\hat{L}_S^{\text{x-e}}]$ by setting the underlying loss function to be the 01 (classification) loss or the cross-entropy loss, respectively. This estimation is valid with high probability (of at least $1-\delta'$) over random weight samples.

The latter expression also can be combined with any of the PAC-Bayes bounds presented in Section 3 to upper-bound the loss $Q_S[L]$ by a computable expression. Just to illustrate, combining with the classical PAC-Bayes bound we would get the following risk bound:

$$Q_S[L] \leq f^\star\Big(\hat{Q}_m[\hat{L}_S], \frac{1}{m}\log(\frac{2}{\delta'})\Big) + \sqrt{\frac{\mathrm{KL}(Q_S\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}$$

which holds with probability at least $1-\delta-\delta'$ over random size-n data samples S and size-m weight samples $W_1,\ldots,W_m \sim Q_S$. The parameter $\delta \in (0,1)$ quantifies the confidence over random data samples, and $\delta' \in (0,1)$ the confidence over random weight samples.

As we said before, our evaluation of risk certificates was based on the PAC-Bayes-kl bound. The next subsection fills the details.

6.2 Final expression for evaluating the risk certificate

In our experiments we evaluate the risk certificates (risk upper bounds) for the cross-entropy loss ($\ell^{\text{x-e}}$) and the 0-1 loss (ℓ^{01}), respectively, computed using the PAC-Bayes-kl bound and Monte Carlo weight sampling. For any $\delta, \delta' \in (0,1)$, with probability at least $1-\delta-\delta'$ over random size-n data samples S and size-m weight samples $W_1, \dots, W_m \sim Q_S$ we have:

$$Q[L] \leq f^{\star}\Bigg(f^{\star}\Big(\hat{Q}_m[\hat{L}_S], \frac{1}{m}\log(\frac{2}{\delta'})\Big), \frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n}\Bigg).$$

In our experiments we used a numerical implementation of the kl inversion $f^{\star}$ and the upper bound just shown to evaluate risk certificates for the stochastic predictors corresponding to the distributions over weights obtained by our training methods.

7. Experimental results

We performed a series of experiments on MNIST and CIFAR-10 to thoroughly investigate the properties of the training objectives presented before. Specifically, we empirically evaluate the two proposed training objectives f_{quad} and f_{lambda} of Eq. (9) and Eq. (10), and compare these to f_{classic} of Eq. (11) and f_{bbb} of Eq. (8). When possible, we also compare to empirical risk minimisation with dropout (f_{erm}). In all experiments, training objectives are compared under the same conditions, i.e. weight initialisation, prior, optimiser (vanilla SGD with momentum) and network architecture. The code for our experiments is publicly available[11] in PyTorch.

7.1 Prior distribution over weights

We studied Gaussian and Laplace distributions over the model weights. The posterior distribution Q is the same kind as the prior in each case.

We also tested in our experiments both data-free random priors (with randomness in the initialization of the weights) and data-dependent priors. In both cases, the center parameters of the prior were initialised randomly from a truncated centered Gaussian distribution with standard deviation set to $1/\sqrt{n_{\text{in}}}$, where n_{in} is the dimension of the inputs to a particular layer, truncating at ± 2 standard deviations. The main difference between our data-free and data-dependent priors is that, after initialisation, the center parameters of data-dependent priors are optimised through ERM on a subset of the training data (50% if not indicated otherwise). This means the posterior center μ will be initialised at the empirical risk minimiser. As opposed to data-dependent priors, in the case of data-free priors we simply use the initial random weights as center parameters. After choosing the center parameter of the prior, the scale parameters ρ is set to the constant scale hyper-parameter. The posterior Q is always initialised at the prior (both center and scale parameters). We find in our experiments that the prior can be over-fitted easily. To avoid this, we use dropout during the learning process (exclusive to learning the prior, not the posterior).

[11] Code available at https://github.com/mperezortiz/PBB

7.2 Experimental setup

All risk certificates were computed using the the PAC-Bayes-kl theorem, as explained in Section 6, with $\delta = 0.025$ and $\delta' = 0.01$ and $m = 150.000$ Monte Carlo model samples (as done by Dziugaite and Roy (2017)). The same confidence δ was used in all the PBB training objectives (f_{quad}, f_{lambda}, f_{classic}). Input data was standardised.

7.2.1 Hyperparameter selection

For all experiments we performed a grid search over all hyper-parameters and selected the run with the best risk certificate on 0-1 error[12] (evaluated as explained in Section 6). We elaborate more on the use of PAC-Bayes bounds for model selection in the next subsection. We did a grid sweep over the prior distribution scale hyper-parameter (i.e. standard deviation) with values in $[0.05, 0.04, 0.03, 0.02, 0.01, 0.005]$. We observed that higher variance values lead to instability during training and lower variance does not explore the weight space. For the SGD with momentum optimizer we performed a grid sweep over learning rate in $[1\text{e}-3, 5\text{e}-3, 1\text{e}-2]$ and momentum in $[0.95, 0.99]$. We found that learning rates higher than $1\text{e}-2$ caused divergence in training and learning rates lower than $5\text{e}-3$ converged slowly. We also found that the best optimiser hyper-parameters for building the data-dependent prior differ from those selected for optimising the posterior. Because of this, we also performed a grid sweep over the learning rate and momentum used for learning the data-dependent prior (testing the same values as before). The dropout rate used for learning the prior was selected from $[0.0, 0.05, 0.1, 0.2, 0.3]$. All training objectives derived from PAC-Bayes bounds used the 'bounded cross-entropy' function as surrogate loss during training, for which we enforced boundedness by restricting the minimum probability (see Section 5). We observed that the value $p_{\text{min}} = 1\text{e}-5$ performed well. Values higher than $1\text{e}-2$ distorts the input to loss function and leads to higher training loss. The lambda value in f_{lambda} was initialised to 1.0 (as done by Thiemann et al. (2017)) and optimized using alternate minimization using SGD with momentum, using the same learning rate and momentum than for the posterior optimisation. Notice that f_{bbb} requires an additional sweep over a KL trade-off coefficient, which was done with values in $[1\text{e}-5, 1\text{e}-4, \ldots, 1\text{e}-1]$, see Blundell et al. (2015).

For ERM, we used the same range for optimising the learning rate, momentum and dropout rate. However, given that in this case we do not have a risk certificate we need to set aside some data for validation and hyper-parameter tuning. We set 4% of the data as validation in MNIST (2400 examples) and 5% in the case of CIFAR-10 (2500 examples). This is the first example of how PAC-Bayes bounds could be a good approach for self-certified learning, showing in this case that, as opposed to ERM, PAC-Bayes inspired training objectives do not need a validation or test set.

[12] Note that if we use a total of C hyperparameter combinations, the union bound correction would add up to $\log(C)/300000$ to the PAC-Bayes-kl upper bound. Even with say $C = 42\text{M}$ (forty two million), the value of our risk certificates, computed via kl inversion, will not be impacted significantly. The reader can be assured that we used much less than 42M hyperparameter combinations.

7.2.2 Predictors and metrics reported

For all methods, we compare three different prediction strategies using the final model weights: i) stochastic predictor, randomly sampling fresh model weights for each test example; ii) deterministic predictor, using exclusively the posterior mean; iii) ensemble predictor, as done by Blundell et al. (2015), in which majority voting is used with the predictions of a number of model weight samples, in our case 100. We report the test cross entropy loss (x-e) and 0-1 error of these predictors. We also report a series of metrics at the end of training (train empirical risk using cross-entropy $Q[\hat{L}_S^{\text{x-e}}]$ and 0-1 error $Q[\hat{L}_S^{01}]$ and KL divergence between posterior and prior) and two risk certificates for the stochastic predictor ($\ell^{\text{x-e}}$ for cross-entropy loss and ℓ^{01} for 0-1 loss).

7.2.3 Architectures

For MNIST, we tested both a fully connected neural network (FCN) with 4 layers (including input and output) and 600 units per layer, as well as a convolutional neural network (CNN) with 4 layers (two convolutional, two fully connected). For the latter, we learn a distribution over the convolutional kernels. We trained our models using the standard MNIST dataset split of 60000 training examples and 10000 test examples. For CIFAR-10, we tested three convolutional architectures (one with a total of 9 layers with learnable parameters and the other two with 13 and 15 layers) with standard CIFAR-10 data splits. ReLU activations were used in each hidden layer for both datasets. Both for learning the posterior and the prior, we ran the training for 100 epochs (however we observed that methods converged around 70). We used a training batch size of 250 for all the experiments.

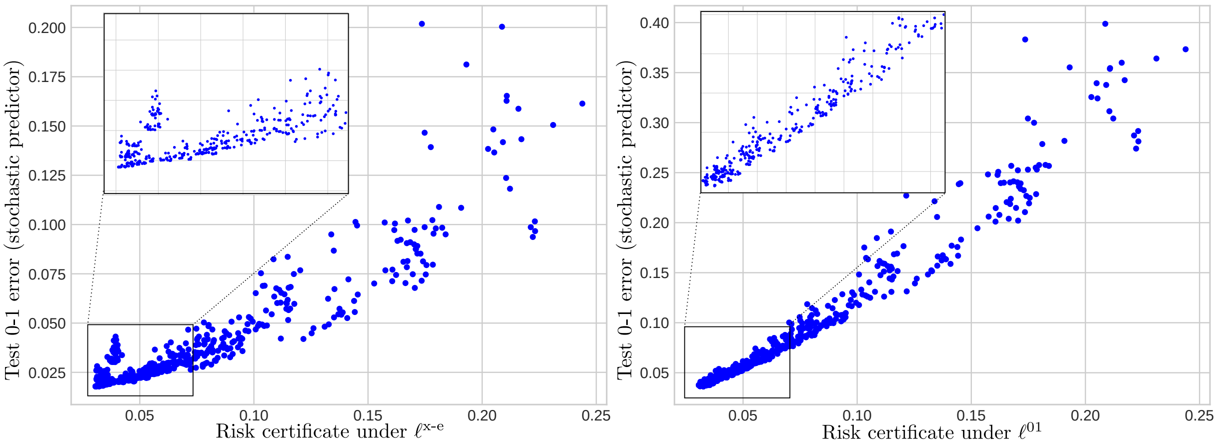

Figure 1: Model selection results from more than 600 runs with different hyper-parameters. The architecture used is a CNN with Gaussian data-dependent priors. We use a reduced subset of MNIST for these experiments (10% of training data).

7.3 Hyper-parameter tuning through PAC-Bayes bounds

We show now that PAC-Bayes bounds can be used not only as training objectives to guide the optimisation algorithm but also for model selection. Specifically, Figure 1 compares the PAC-Bayes-kl bound for cross-entropy and 0-1 losses (x-axis) to the test 0-1 error for the stochastic predictor (y-axis) for more than 600 runs from the hyper-parameter grid search performed for f_{quad} with a CNN architecture and a data-dependent Gaussian prior on MNIST. We do a grid search over 6 hyper-parameters: prior scale, dropout rate, and the learning rate and momentum both for learning the prior and the posterior. To depict a larger range of performance values (thus avoiding only showing the risk and performance for relatively accurate classifiers) we use here a reduced training set for these experiments (i.e. 10% of training data from MNIST). The test set is maintained. The results show a clear positive correlation between the risk certificate and test set 0-1 error, especially for the risk certificate of the 0-1 error, as expected. While the plots also show heterokedasticity (there is a noticeable increase of variability towards the right side of the x-axis) the crucial observation is that for small error values the corresponding values of the risk certificate are reasonable stable. It is worth keeping in mind, however, that bounds generally get weaker with higher error values.

Motivated by the results in the plots, where it is shown that the bound could potentially be used for model selection, we use the risk certificate of the 0-1 loss (evaluated as explained in Section 6) for hyper-parameter tuning in all our subsequent experiments. Note that the advantage in this case is that we do not need a validation set.

Setup			Risk cert.	&	Train metrics			Stch. pred.		Det. pred.		Ens. pred.		Prior
Arch.	Prior	Obj.	$\ell^{\text{x-e}}$	ℓ^{01}	KL/n	$Q[\hat{L}_S^{\text{x-e}}]$	$Q[\hat{L}_S^{01}]$	x-e	01 err.	x-e	01 err.	x-e	01 err.	01 err.
FCN	Rand.Init. (Gaussian)	f_{quad}	.2033	.3155	.1383	.0277	.0951	.0268	.0921	.0137	.0558	.0007	.0572	.8792
		f_{lambda}	.2326	.3275	.1856	.0218	.0742	.0211	.0732	.0077	.0429	.0004	.0448	.8792
		f_{classic}	.1749	.3304	.0810	.0433	.1531	.0407	.1411	.0204	.0851	.0009	.0868	.8792
		f_{bbb}	.5163	.5516	.6857	.0066	.0235	.0088	.0293	.0038	.0172	.0003	.0178	.8792
	Learnt (Gaussian)	f_{quad}	.0146	.0279	.0010	.0092	.0204	.0084	.0202	.0032	.0186	.0002	.0189	.0202
		f_{lambda}	.0201	.0354	.0054	.0073	.0178	.0082	.0196	.0071	.0185	.0001	.0185	.0202
		f_{classic}	.0141	.0284	.0001	.0115	.0247	.0101	.0230	.0089	.0189	.0002	.0191	.0202
		f_{bbb}	.0788	.0968	.0704	.0025	.0090	.0063	.0179	.0066	.0153	.0001	.0153	.0202
	-	f_{erm}	-	-	-	.0004	.0007	-	-	.0101	.0152	-	-	-
CNN	Rand.Init. (Gaussian)	f_{quad}	.1453	.2165	.1039	.0157	.0535	.0143	.0513	.0062	.0257	.0003	.0261	.9478
		f_{lambda}	.1583	.2202	.1256	.0126	.0430	.0109	.0397	.0056	.0207	.0003	.0211	.9478
		f_{classic}	.1260	.2277	.0622	.0273	.0932	.0253	.0869	.0111	.0425	.0006	.0421	.9478
		f_{bbb}	.3400	.3645	.3948	.0034	.0120	.0039	.0154	.0016	.0088	.0001	.0092	.9478
	Learnt (Gaussian)	f_{quad}	.0078	.0155	.0001	.0058	.0127	.0045	.0104	.0003	.0105	.0001	.0104	.0104
		f_{lambda}	.0095	.0186	.0010	.0051	.0123	.0044	.0106	.0047	.0098	.0000	.0100	.0104
		f_{classic}	.0083	.0166	.0000	.0064	.0139	.0049	.0123	.0048	.0103	.0001	.0103	.0104
		f_{bbb}	.0447	.0538	.0398	.0012	.0042	.0040	.0104	.0043	.0082	.0002	.0082	.0104
	-	f_{erm}	-	-	-	.0003	.0004	-	-	.0081	.0092	-	-	-

Table 1: Train and test set metrics on MNIST using Gaussian priors. The table includes two architectures (FCN and CNN), two priors (a data-free prior centered at the randomly initialised weights, and a data-dependent prior learnt on a subset of the dataset) and four training objectives.

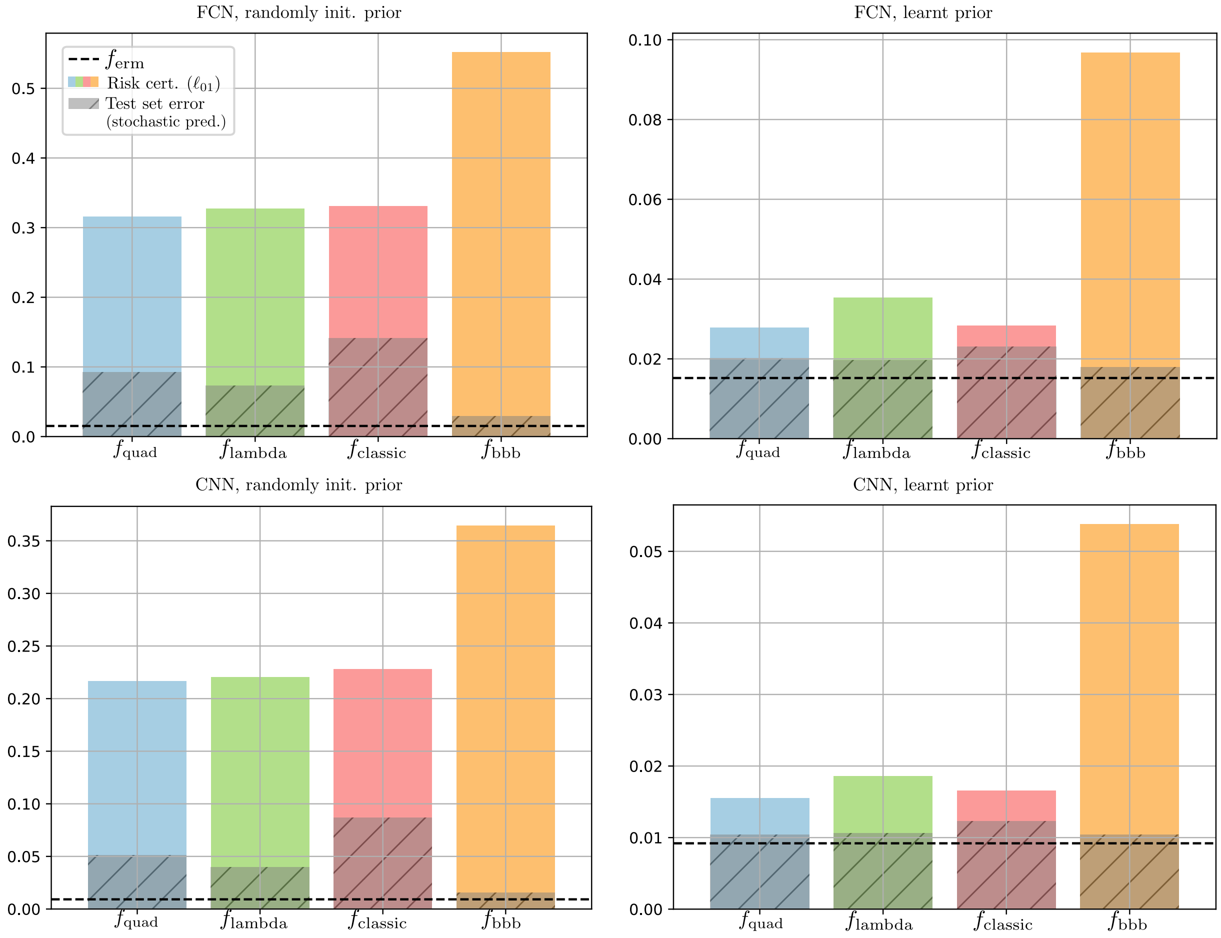

Figure 2: Bar plots of results across different architectures, priors and training objectives for MNIST. The bottom shaded areas correspond to the test set 0-1 error of the stochastic classifier. The coloured areas on top correspond to the risk certificate. The horizontal dashed line corresponds to the test set 01 error of f_{erm}, i.e. the deterministic classifier learnt by empirical risk minimisation of the surrogate loss on the whole training set (shown for comparison purposes).

7.4 Comparison of different training objectives and priors

We first present a comparison of the four considered training objectives on MNIST using Gaussian weight distributions. Table 1 shows the results for the two architectures previously described for MNIST (FCN and CNN) and both data-free and data-dependent priors (referred to as Rand.Init. and Learnt, respectively). We also include the results obtained by standard ERM using the cross-entropy loss, for which part of the table can not be completed (e.g. risk certificates). The last column of the table shows the test 0-1 error of the deterministic prior predictor. We also report the test performance for the stochastic predictor (column named Stch. pred.), the posterior mean deterministic predictor (column named Det. pred.) and the ensemble predictor (column named Ens pred.). An important note is that we used the risk certificates for model selection for all training objectives, including f_{bbb} (with the sole exception of f_{erm}). The KL trade-off coefficient included in f_{bbb} (Blundell et al., 2015) relaxes the importance given to the prior in the optimisation, but obviously not in the computation of the risk certificate, which in practice means that larger KL attenuating coefficients will lead to worse risk certificates. Because of this, in all cases, the model selection strategy chose the lowest value for the KL attenuating coefficient (0.1) for f_{bbb}, meaning there are cases in which f_{bbb} obtained better test performance than the ones we report in this table, but much looser risk certificates. We present more experiments on this in the next subsection where we experiment with the KL attenuating trick.

The findings from the experiments in Table 1 and Figure 2 are as follows: i) f_{quad} achieves consistently the best risk certificates for 0-1 error (see ℓ_{01}) in all experiments, providing as well better test performance than f_{classic}, as observed when comparing the 0-1 loss of the stochastic predictors. ii) Based on the results of the stochastic predictor, f_{lambda} is the best PAC-Bayes inspired objective in terms of test performance, although the risk certificates are generally less tight. iii) In most cases, the stochastic predictor does not worsen the performance of the prior predictor, improving it very significantly for random data-free priors (i.e. Rand.Init). iv) The mean of the weight distribution is also improved, as shown by comparing the results of the deterministic predictor (Det. pred.) to the prior predictor. The ensemble predictor also generally improves on the prior. v) The improvements brought by data-dependent priors (labelled as "Learnt" in the table) are consistent across the two architectures, showing better test performance and risk certificates (although the use of data-free priors still produced non-vacuous risk certificates). vi) The application of PBB is successful not only for learning fully connected layers but also for learning convolutional ones. The improvements in performance and risk certificates that the use of a CNN brings are also noteworthy. vii) The proposed PAC-Bayes inspired learning strategies show competitive performance when compared to state-of-the-art f_{bbb} and f_{erm} (specially when using data-dependent priors) while providing tight risk certificates.

We now compare our results to those reported before in the literature for MNIST. Note that in this case there are differences regarding optimiser, prior chosen and weight initialisation (however, the neural network architecture used is the same, FCN as described in this paper). Dziugaite and Roy (2018) implemented a version of f_{classic} and the bound of Lever et al. (2013) for comparison. We compare the results reported by them with the results of training with our two training objectives f_{quad} and f_{lambda}, and with f_{classic} (optimized as per our f_{quad} and f_{lambda}). These results are presented in Table 2. The hyperparameter τ in both Dziugaite and Roy (2018) and Lever et al. (2013) controls the temperature of a Gibbs distribution with unnormalized density $e^{-\tau \hat{L}_S(w)}$ with respect to some fixed measure on weight space. In the table we display only the two values of their τ parameter which achieve best test error and risk certificate. We note that Dziugaite and Roy (2018)'s best values correspond to test accuracy of 94% or 93% while in those cases their risk certificates (0.650 or 0.350, respectively), although non-vacuous, were far from tight. On the other hand the tightest value of their risk bound (0.21) only gives an 88% accuracy. In contrast, our PBB methods achieve close to 98% test accuracy (or 0.0202 test error). At the same time, as noted above, our risk certificate (0.0279) is much tighter

Method	Stch. Pred. 01 Err	Risk certificate
D&R 2018 ($\tau = 3e+3$)	0.120	0.2100
D&R 2018 ($\tau = 1e+5$)	0.060	0.6500
Lever et al. 2013 ($\tau = 3e+3$)	0.120	0.2600
Lever et al. 2013 ($\tau = 1e+5$)	0.060	1.0000
pb_quad	0.0202	0.0279
pb_lambda	0.0196	0.0354
pb_classic	0.0230	0.284

Table 2: Comparison of test set 0-1 error for the stochastic predictor and risk certificate for standard MNIST dataset. We compare here our results for the FCN with data-dependent priors to previous published work. All methods use data-dependent priors (albeit different ones) and exactly the same architecture.

than theirs (0.210), meaning that our training scheme (not only training objectives but also prior) are a significant improvement with respect to theirs (an order of magnitude tighter). Even more accurate predictors and tighter bounds are achieved by the CNN architecture, as shown in Table 1.

7.5 KL attenuating trick

As many works have pointed out before (and we have observed in our experiments), the problem with all the four presented training objectives is that the KL term tends to dominate and most of the work in training is targeted at reducing it, which effectively means often the posterior cannot move far from the prior. To address this issue, distribution-dependent (Lever et al. (2013)) or data-dependent (Dziugaite and Roy (2018)) priors have been used in the literature. Another approach to address this is to add a coefficient that controls the influence of the KL in the training objective (Blundell et al., 2015). This means that in the case of f_{bbb} we could see marginal decrease in the KL divergence during the course of training (specially given small KL attenuating coefficients) and the solution it returns is expected to be similar to that returned simply using ERM with cross-entropy. However, this also has its effects on the risk certificate. To show these effects, we run all four training objectives with a KL penalty of 0.0001 during training and report the results in Table 3. For simplicity, only a CNN architecture is considered in this experiment. What we can see comparing these results to the ones reported in Table 1 is that while the 0-1 error for the stochastic classifier decreases, the KL term increases and so does the final risk certificate. Practitioners may want to consider this trade-off between performance and tight risk certificates.

7.6 Laplace weight distributions

We experimented with both Laplace and Gaussian priors. The results are presented in Table 4. Comparing these to the results with Gaussian weight distributions from Table 1, we did not observe significant and consistent differences in terms of risk certificates and test set error between the two priors. The distribution to use could be problem-dependent, but

Setup		Risk cert.	&	Train metrics			Stch. pred.		Det. pred.		Ens. pred.		Prior
Arch. & Prior	Obj.	$\ell^{\text{x-e}}$	ℓ^{01}	KL/n	$Q[\hat{L}_S^{\text{x-e}}]$	$Q[\hat{L}_S^{01}]$	x-e	01 err.	x-e	01 err.	x-e	01 err.	01 err.
CNN Rand.Init (KL attenuating)	f_{quad}	.2292	.2824	.2174	.0097	.0330	.0084	.0305	.0042	.0193	.0002	.0201	.9478
	f_{lambda}	.2840	.3241	.3004	.0066	.0225	.0058	.0222	.0039	.0144	.0002	.0148	.9478
	f_{classic}	.2297	.2846	.2167	.0101	.0344	.0096	.0343	.0047	.0208	.0002	.0216	.9478
	f_{bbb}	.4815	.4974	.6402	.0024	.0082	.0035	.0107	.0024	.0082	.0000	.0079	.9478
CNN Learnt (KL attenuating)	f_{quad}	.0191	.0296	.0104	.0030	.0087	.0033	.0101	.0000	.0095	.0000	.0096	.0104
	f_{lambda}	.0245	.0354	.0162	.0025	.0076	.0031	.0092	.0040	.0092	.0000	.0095	.0104
	f_{classic}	.0187	.0296	.0100	.0031	.0089	.0037	.0106	.0043	.0095	.0001	.0095	.0104
	f_{bbb}	.0470	.0557	.0421	.0012	.0041	.0034	.0096	.0025	.0085	.0001	.0083	.0104

Table 3: Train and test set results on MNIST using Gaussian priors and a penalty on the KL of 0.001 for all training objectives. Only a CNN architecture is considered.

we found that both Gaussian and Laplace distributions achieve good risk certificates and test set performance.

Setup			Risk cert.	&	Train metrics			Stch. pred.		Det. pred.		Ens. pred.		Prior
Arch.	prior	Obj.	$\ell^{\text{x-e}}$	ℓ^{01}	KL/n	$Q[\hat{L}_S^{\text{x-e}}]$	$Q[\hat{L}_S^{01}]$	x-e	01 err.	x-e	01 err.	x-e	01 err.	01 err.
CNN	Rand.Init. (Laplace)	f_{quad}	.1548	.2425	.1024	.0207	.0709	.0190	.0677	.0113	.0429	.0004	.0436	.9478
		f_{lambda}	.1844	.2540	.1489	.0147	.0496	.0131	.0461	.0096	.0310	.0003	.0312	.9478
		f_{classic}	.1334	.2489	.0610	.0322	.1101	.0296	.1014	.0208	.0719	.0007	.0695	.9478
		f_{bbb}	.4280	.4487	.5385	.0031	.0107	.0038	.0139	.0006	.0096	.0001	.0090	.9478
	Learnt (Laplace)	f_{quad}	.0085	.0167	.0004	.0056	.0126	.0043	.0098	.0011	.0103	.0001	.0103	.0104
		f_{lambda}	.0119	.0216	.0025	.0049	.0118	.0041	.0106	.0052	.0103	.0003	.0100	.0104
		f_{classic}	.0076	.0155	.0000	.0060	.0131	.0046	.0107	.0015	.0105	.0001	.0106	.0104
		f_{bbb}	.0737	.0866	.0673	.0019	.0062	.0031	.0092	.0013	.0093	.0001	.0091	.0104

Table 4: Train and test set results on MNIST using Laplace priors. For simplicity, only a CNN architecture is considered here.

Figure 3 shows a summary of all the results obtained for MNIST (i.e. results reported in Table 1 and Table 4). This shows clearly the differences between the three training objectives: f_{lambda} tends to lead generally to the lowest test set error, but worse risk certificates than f_{quad}, and f_{classic} leads to the worse test set performance and looser bounds. Thus, f_{quad} gives a reasonable trade-off between test set performance and tight risk certificates. The general trend of the relationship shows a slight curvature, as also seen in Figure 1.

7.7 CIFAR-10 with larger architectures

We evaluate now our training objectives on CIFAR-10 using deep CNN architectures. Note that this is a much larger scale experiment than the ones presented before (15 layers with learnable parameters vs 4). As far as we know, we are the first to evaluate PAC-Bayes inspired training objectives in such deep architectures. The results are presented in Table 5 and Figure 4 for three architectures (with 9, 13 and 15 layers, with around 6M, 10M and 13M parameters, respectively). Note, however, that the number of parameters is doubled for our probabilistic neural networks. We also experiment with using different amount of data for learning the prior: 50% and 70%, leaving respectively 25.000 and 15.000 examples to evaluate the bound. The conclusions are as follows: i) In this case, the improvements brought by learning the posterior through PBB with respect to the prior are much better

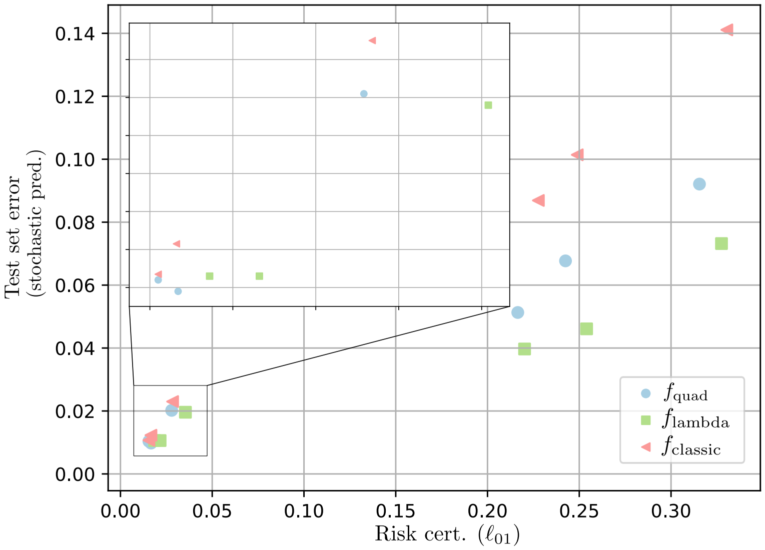

Figure 3: Scatter plot of the results obtained for MNIST using different training objectives. The x-axis represents the risk certificate on the 01 loss and the y-axis the test set 01 loss achieved by the stochastic classifier.

and generally consistent across all experiments (e.g. 2 points in test 0-1 error for f_{lambda} when using 50% of the data for learning the prior). ii) Risk certificates are also non-vacuous and tight (although less than for MNIST). iii) We validate again that f_{lambda} shows better test performance but less tight risk certificates. iv) In this case, however, f_{classic} and f_{quad} seem much closer in terms of performance and tightness. In some cases, f_{classic} provides slightly tighter bounds, but also often worse test performance. The tighter bounds can be explained by our findings with the Pinsker inequality, which makes f_{classic} tighter when true loss is more than 0.25. This observation can be seen clearly in Figure 5. v) Obtained results with 15 layers are competitive, achieving similar performance than those reported in the state-of-the-art for VGG-16 (deep network proposed for CIFAR-10 with 16 layers). vi) The results indicate that 50% of the training data is not enough in this dataset to build a competitive prior and this influences the test performance and the risk certificates. The results with 70% of the data are, however, very close to those achieved by ERM across all three architectures. vii) Similarly than with the rest of the experiments, a major difference can be seen when comparing the risk certificate achieved by f_{bbb} with the risk certificate achieved by PAC-Bayes inspired training objectives. viii) Finally, it is noteworthy how the KL gets generally smaller as we move to deeper architectures (specially from 9 to 13 layers), which is an interesting observation, as there are many more parameters used in the computation of the KL. This indicates that the posterior in deeper architectures stays much closer to the prior. We believe this may be because in a higher-dimensional weight space, the weight updates have a smaller euclidean norms, hence the smaller KL.

Finally, we compare the test set performance of the different predictors considered in this work (stochastic, deterministic and ensemble). The results for MNIST and CIFAR-

Setup			Risk cert. & Train metrics					Stch. pred.		Det. pred.		Ens. pred.		Prior
Arch.	Prior	Obj.	$\ell^{\text{x-e}}$	ℓ^{01}	KL/n	$Q[\hat{L}_S^{\text{x-e}}]$	$Q[\hat{L}_S^{01}]$	x-e	01 err.	x-e	01 err.	x-e	01 err.	01 err.
CNN (9 layers)	Learnt (50% data)	f_{quad}	.1296	.3034	.0089	.0868	.2428	.0903	.2452	.0726	.2439	.0024	.2413	.2518
		f_{lambda}	.1742	.3730	.0611	.0571	.2108	.0689	.2307	.0609	.2225	.0018	.2133	.2518
		f_{classic}	.1173	.2901	.0035	.0903	.2511	.0931	.2537	.0952	.2437	.0025	.2332	.2518
		f_{bbb}	.8096	.8633	1.5107	.0239	.0926	.0715	.2198	.0735	.2160	.0017	.2130	.2518
	Learnt (70% data)	f_{quad}	.1017	.2502	.0026	.0796	.2179	.0816	.2137	.0928	.2137	.0023	.2100	.2169
		f_{lambda}	.1414	.3128	.0307	.0630	.2022	.0708	.2081	.0767	.2061	.0021	.2049	.2169
		f_{classic}	.0957	.2377	.0004	.0851	.2223	.0862	.2161	.0827	.2167	.0021	.2135	.2169
		f_{bbb}	.6142	.6965	.8397	.0212	.0822	.0708	.1979	.0562	.1992	.0019	.1944	.2169
	-	f_{erm}	-	-	-	.0355	.0552	-	-	.1400	.1946	-	-	-
CNN (13 layers)	Learnt (50% data)	f_{quad}	.0821	.2256	.0042	.0577	.1874	.0585	.1809	.0519	.1788	.0011	.1783	.1914
		f_{lambda}	.1163	.2737	.0272	.0491	.1741	.0516	.1740	.0466	.1726	.0015	.1690	.1914
		f_{classic}	.0757	.2127	.0009	.0635	.1936	.0622	.1880	.0592	.1810	.0017	.1816	.1914
		f_{bbb}	.6787	.7566	.9999	.0250	.0924	.0505	.1676	.0422	.1646	.0011	.1614	.1914
	Learnt (70% data)	f_{quad}	.0659	.1832	.0015	.0519	.1608	.0517	.1568	.0421	.1553	.0010	.1546	.1587
		f_{lambda}	.0896	.2177	.0145	.0449	.1499	.0479	.1541	.0604	.1522	.0011	.1507	.1587
		f_{classic}	.0619	.1758	.0002	.0548	.1644	.0541	.1588	.0605	.1578	.0013	.1557	.1587
		f_{bbb}	.4961	.5858	.5826	.0213	.0772	.0487	.1508	.0532	.1495	.0016	.1461	.1587
	-	f_{erm}	-	-	-	.0576	.0810	-	-	.0930	.1566	-	-	-
CNN (15 layers)	Learnt (50% data)	f_{quad}	.0867	.2174	.0053	.0587	.1753	.0584	.1668	.0538	.1662	.0014	.1653	.1688
		f_{lambda}	.1217	.2707	.0304	.0494	.1661	.0506	.1618	.0417	.1639	.0015	.1622	.1688
		f_{classic}	.0782	.1954	.0007	.0667	.1783	.0652	.1686	.0594	.1692	.0013	.1674	.1688
		f_{bbb}	.6069	.7066	.7908	.0287	.1073	.0468	.1553	.0412	.1530	.0012	.1517	.1688
	Learnt (70% data)	f_{quad}	.0756	.1806	.0028	.0559	.1513	.0559	.1463	.0391	.1469	.0016	.1449	.1490
		f_{lambda}	.0922	.2121	.0133	.0486	.1477	.0500	.1437	.0507	.1449	.0012	.1438	.1490
		f_{classic}	.0703	.1667	.0003	.0622	.1548	.0615	.1475	.0551	.1480	.0010	.1476	.1490
		f_{bbb}	.4481	.5572	.4795	.0259	.0947	.0455	.1413	.0395	.1405	.0008	.1409	.1490
	-	f_{erm}	-	-	-	.0208	.0339	-	-	.0957	.1413	-	-	-

Table 5: Train and test set results on CIFAR-10 using Gaussian priors, three deep CNN architectures and two percentages of data used to build the data-dependent prior (50% and 70%, i.e. 25.000 and 35.000 examples respectively).

10 are depicted in Figure 6. One can appreciate a very clear linear relationship between predictors. In the case of CIFAR-10 the results are similar across all predictors, whereas for MNIST the stochastic predictor obtains significantly worse results (see differences in scales of x and y axes). In the case of CIFAR-10 this may hint that or training strategy finds a solution within a large region of comparably good solutions, so that weight randomization does not affect significantly the test performance of the classifier. We plan to explore this interesting phenomenon in future work.

8. Conclusion and future work

In this paper we explored 'PAC-Bayes with Backprop' (PBB) methods to train probabilistic neural networks with different weight distributions, priors and network architectures. The take-home message is that the training methods presented in this paper are derived from sound theoretical foundations and provide a simple strategy that comes with a performance guarantee at a relatively low extra cost in performance. This is an improvement over methods derived heuristically rather than from theoretically justified arguments, and over methods that do not include a risk certificate valid on unseen examples. Additionally, we

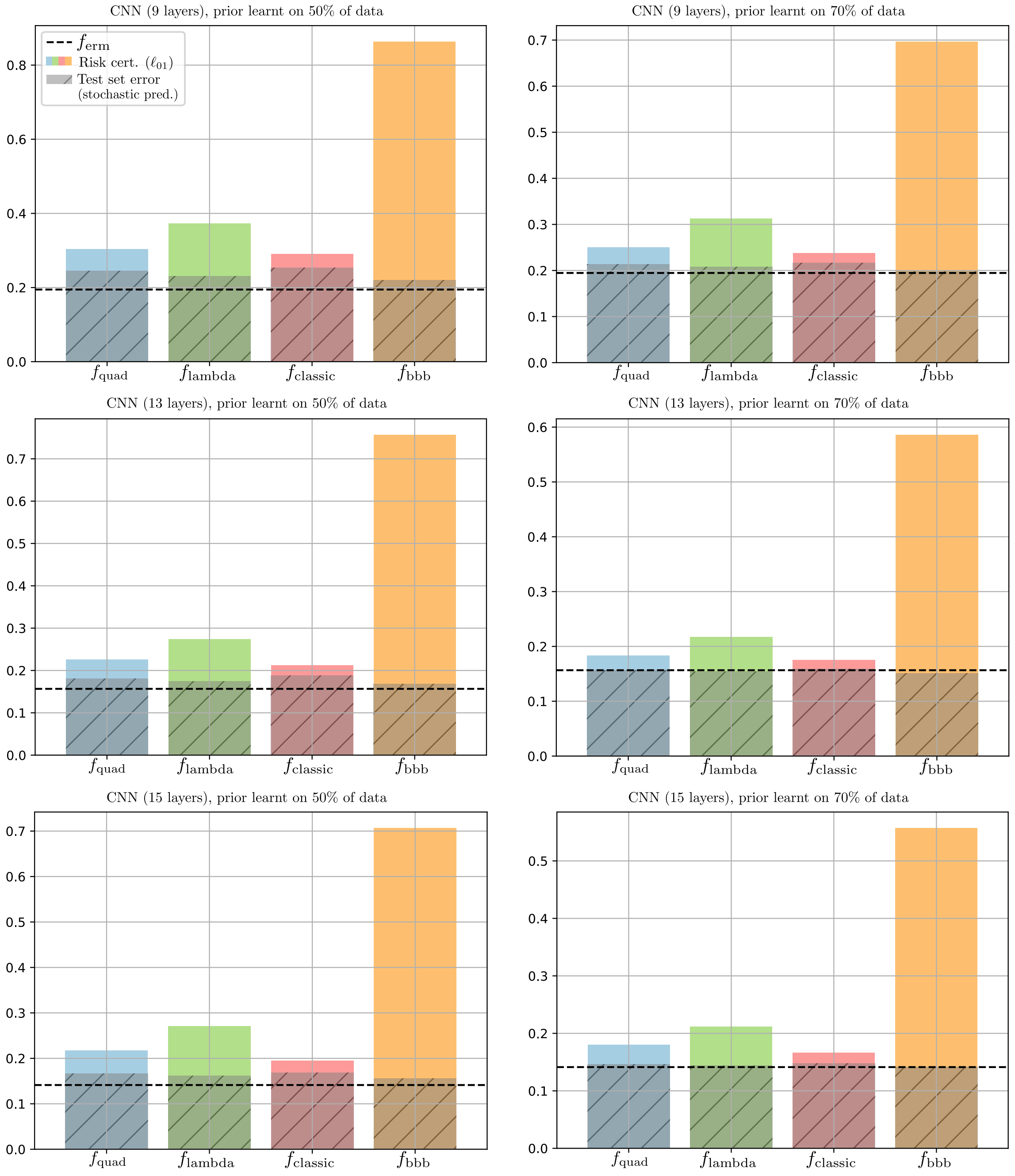

Figure 4: Bar plots of results achieved on CIFAR-10 for 3 different network architectures and two data-dependent priors (learnt using 50% and 70% of the data).

empirically demonstrate the usefulness of data-dependent priors for achieving competitive test performance and, importantly, for computing risk certificates with tight values.

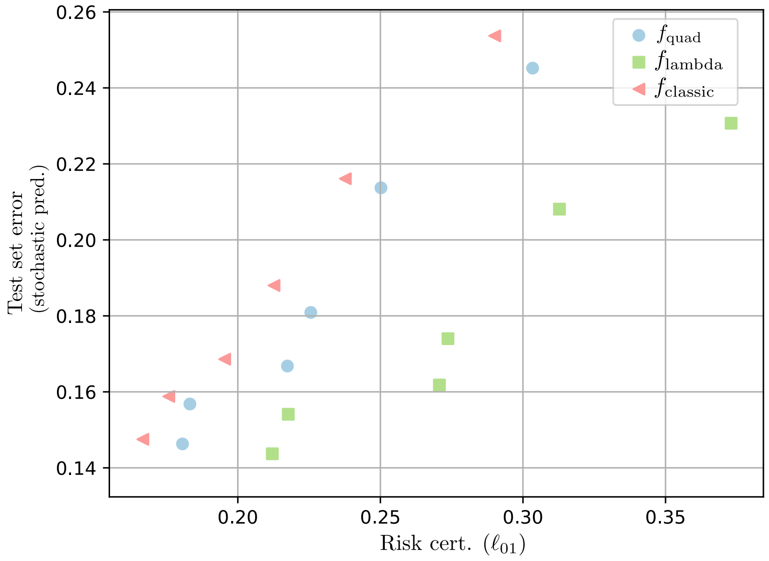

Figure 5: Scatter plot of the results obtained for CIFAR-10 using three different training objectives. The x-axis represents the risk certificate on the 01 loss and the y-axis the test set 01 loss achieved by the stochastic classifier.

The results of our experiments on MNIST and CIFAR-10 have showed that these PBB objectives give predictors with competitive test set performance and with non-vacuous risk certificates that significantly improve previous results and can be used not only for guiding the learning algorithm and certifying the risk but also for model selection. This shows that PBB methods are promising examples of self-certified learning, since the values of the risk certificates output by the training methods are tight, i.e. close to the values of the test set error estimates. We also evaluated our training objectives on large convolutional neural networks (up to 15 layers and around 13M parameters). Our results showed risk certificates with values not as close to the test set error estimate as in the MNIST experiments but still non-vacuous and relatively tight (18% of risk certificate for a stochastic predictor that achieves 14.6% of test 0-1 error). Note that to claim that self-certified learning is achieved would require either testing a given training method across a wide range or datasets and architectures (so as to experimentally validate the claim), or theoretically characterizing the problems on which a given learning method will produce tight risk certificates.

In future work we plan to test different covariance structures for the weight distribution and validate a more extensive list of choices for the weight distributions across a larger list of datasets. We also plan to experiment how to approach the well-known dominance of the KL term in the optimisation of these objectives. Data-dependent priors seem like a promising avenue to do so. We will also explore deeper architectures. Finally, we plan to study risk certificates for the ensemble predictor. We also plan to study different ensemble methods, for instance the one that Thiemann et al. (2017) used with SVMs looks promising, it would be interesting to explore such method (and others) with neural networks.

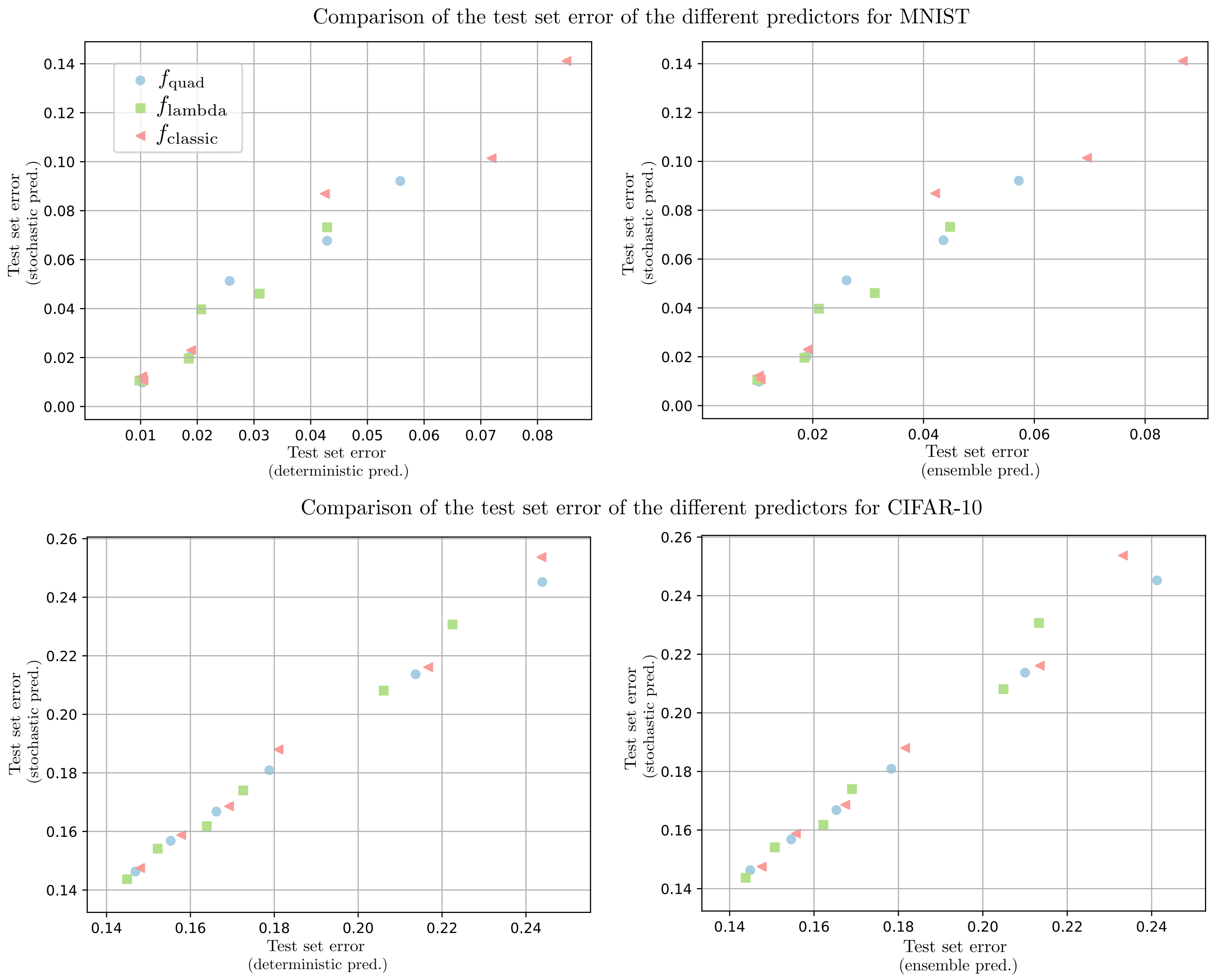

Figure 6: Representation of the results achieved by the different predictors that were studied (stochastic, deterministic and ensemble).

References

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

Charu C. Aggarwal. *Neural networks and deep learning*. Springer, 2018.

David Barber and Christopher M Bishop. Ensemble learning for multi-layer networks. In *Advances in neural information processing systems*, pages 395–401, 1998.

Felix Biggs and Benjamin Guedj. Differentiable PAC-Bayes Objectives with Partially Aggregated Neural Networks. arXiv:2006.12228, 2020.

Pier Giovanni Bissiri, Chris C Holmes, and Stephen G Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130, 2016.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *32nd International Conference on Machine Learning*, pages 1613–1622, 2015.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

Wray L Buntine and Andreas S Weigend. Bayesian back-propagation. *Complex systems*, 5 (6):603–643, 1991.

Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems*, pages 402–408, 2001.

Olivier Catoni. PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning. arXiv:0712.0248, 2007.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, pages 2350–2358, 2015a.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 117–126. ACM, 2015b.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. In *UAI*, 2017.

Gintare Karolina Dziugaite and Daniel M Roy. Data-dependent PAC-Bayes priors via differential privacy. In *Advances in Neural Information Processing Systems*, pages 8430–8441, 2018.

Gintare Karolina Dziugaite, Kyle Hsu, Waseem Gharbieh, and Daniel M Roy. On the role of data in PAC-Bayes bounds. arXiv:2006.10929, 2020.

Charles W Fox and Stephen J Roberts. A tutorial on variational Bayesian inference. *Artificial Intelligence Review*, 38(2):85–95, 2012.

Yoav Freund. Self bounding learning algorithms. In *Proceedings of the eleventh annual conference on Computational Learning Theory*, pages 247–258. ACM, 1998.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *Proc. of the 26th International Conference on Machine Learning*, pages 353–360. ACM, 2009.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Benjamin Guedj. A Primer on PAC-Bayesian Learning. arXiv:1901.05353, 2019.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

Geoffrey E Hinton and Drew van Camp. Keeping neural networks simple. In *International Conference on Artificial Neural Networks*, pages 11–18. Springer, 1993.

Martin Jankowiak and Fritz Obermeyer. Pathwise Derivatives Beyond the Reparameterization Trick. arXiv:1806.01851, 2018.

Joseph Keshet, David McAllester, and Tamir Hazan. PAC-Bayesian approach for minimization of phoneme error rate. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2224–2227. IEEE, 2011.

Joseph Keshet, Subhransu Maji, Tamir Hazan, and Tommi Jaakkola. Perturbation Models and PAC-Bayesian Generalization Bounds. In *Perturbations, Optimization, and Statistics*, pages 289–309. MIT Press, 2017. URL http://u.cs.biu.ac.il/~jkeshet/papers/KeshetMaHaJa16.pdf.

Xinjie Lan, Xin Guo, and Kenneth E Barner. PAC-Bayesian Generalization Bounds for MultiLayer Perceptrons. arXiv:2006.08888, 2020.

John Langford and Avrim Blum. Microchoice bounds and self bounding learning algorithms. *Machine Learning*, 51(2):165–179, 2003.

John Langford and Rich Caruana. (Not) bounding the true error. In *Advances in Neural Information Processing Systems*, pages 809–816, 2001.

John Langford and Matthias Seeger. Bounds for averaging classifiers. Technical Report CMU-CS-01-102, Carnegie Mellon University, 2001.

Gaël Letarte, Pascal Germain, Benjamin Guedj, and François Laviolette. Dichotomize and generalize: PAC-Bayesian binary activated deep neural networks. In *Advances in Neural Information Processing Systems*, pages 6872–6882, 2019.

Guy Lever, François Laviolette, and John Shawe-Taylor. Tighter PAC-Bayes bounds through distribution-dependent priors. *Theoretical Computer Science*, 473:4–28, 2013.

Ben London. A PAC-Bayesian analysis of randomized learning with application to stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 2931–2940, 2017.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016.

David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

Andreas Maurer. A note on the PAC Bayesian theorem. arXiv:cs/0411099, 2004.

David A McAllester. PAC-Bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational Learning Theory*, pages 164–170. ACM, 1999.

David A McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51(1): 5–21, 2003.

Grégoire Montavon, Geneviève Orr, and Klaus-Robert Müller. *Neural networks: tricks of the trade*, volume 7700. springer, 2012.

Radford M Neal. Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Technical Report CRG-TR-92-1, University of Toronto, 1992.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

Asaf Noy and Koby Crammer. Robust forward algorithms via PAC-Bayes and Laplace distributions. In *Artificial Intelligence and Statistics*, pages 678–686, 2014.

Emilio Parrado-Hernández, Amiran Ambroladze, John Shawe-Taylor, and Shiliang Sun. Pac-bayes bounds with data dependent priors. *Journal of Machine Learning Research*, 13(112):3507–3531, 2012. URL http://jmlr.org/papers/v13/parrado12a.html.

Robert Price. A useful theorem for nonlinear devices having Gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable Laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

Omar Rivasplata, Ilja Kuzborskij, Csaba Szepesvári, and John Shawe-Taylor. PAC-Bayes Analysis Beyond the Usual Bounds. NeurIPS 2019 Workshop on Machine Learning with Guarantees, 2019a.

Omar Rivasplata, Vikram M Tankasali, and Csaba Szepesvári. PAC-Bayes with Backprop. arXiv:1908.07380, 2019b.

Francisco R Ruiz, Michalis Titsias, and David Blei. The Generalized Reparameterization Gradient. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 460–468, 2016.

Matthias Seeger. PAC-Bayesian Generalization Error Bounds for Gaussian Process Classification. *Journal of Machine Learning Research*, 3:233–269, 2002.

Yevgeny Seldin and Naftali Tishby. PAC-Bayesian analysis of co-clustering and beyond. *Journal of Machine Learning Research*, 11(Dec):3595–3646, 2010.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning.* Cambridge University Press, Cambridge, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Ingo Steinwart and Andreas Christmann. *Support Vector Machines.* Springer, 2008.

Sanjay Thakur, Herke Van Hoof, Gunshi Gupta, and David Meger. Unifying Variational Inference and PAC-Bayes for Supervised Learning that Scales. arXiv:1910.10367, 2019.

Niklas Thiemann, Christian Igel, Olivier Wintenberger, and Yevgeny Seldin. A strongly quasiconvex PAC-Bayesian bound. In *International Conference on Algorithmic Learning Theory*, pages 466–492, 2017.

Ilya O Tolstikhin and Yevgeny Seldin. PAC-Bayes-empirical-Bernstein inequality. In *Advances in Neural Information Processing Systems*, pages 109–117, 2013.

Tim van Erven. PAC-Bayes Mini-tutorial: A Continuous Union Bound. arXiv:1405.1580, 2014.

Paul Viallard, Rémi Emonet, Pascal Germain, Amaury Habrard, and Emilie Morvant. Interpreting Neural Networks as Majority Votes through the PAC-Bayesian Theory. NeurIPS 2019 Workshop on Machine Learning with Guarantees, 2019.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan Adams, and Peter Orbanz. Non-vacuous generalization bounds at the ImageNet scale: A PAC-Bayesian compression approach. In *International Conference on Learning Representations*, 2019.