Towards self-certified learning: Probabilistic neural networks trained by PAC-Bayes with Backprop

María Pérez-Ortiz
AI Centre, University College London
maria.perez@ucl.ac.uk

Omar Rivasplata
DeepMind
rivasplata@google.com

John Shawe-Taylor
AI Centre, University College London
jst@cs.ucl.ac.uk

Csaba Szepesvári
DeepMind
szepi@google.com

Abstract

The result of training a probabilistic neural network is a probability distribution over network weights. This learnt distribution is the basis of a prediction scheme, e.g. building a stochastic predictor or integrating the predictions of all possible parameter settings. In this paper we experiment with training 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. We show that the methods studied here represent promising candidates for self-certified learning, achieving state-of-the-art test performance in several data sets and at the same time obtaining reasonably tight certificates on the risk on any unseen data without the need for data-splitting protocols (both for testing and model selection).

1 Introduction

This paper is largely based on our previous work in Pérez-Ortiz et al. [2020]. We present empirical studies on training probabilistic neural networks from a PAC-Bayesian approach. The generic method, which we call 'PAC-Bayes with Backprop' (PBB), consists of minimizing a PAC-Bayes bound through stochastic gradient descent. Our work takes inspiration from Blundell et al. [2015] whose method (called Bayes-by-Backprop) showed that randomized weights achieve competitive test set error; and from Dziugaite and Roy [2017, 2018], whose results showed that randomized neural networks obtained by minimizing a classic PAC-Bayes bound can not only have reasonable test set errors but also, more importantly, non-vacuous risk bound values.

A clear advantage of PBB methods is being an instance of self-certified[1] learning. 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 (cf. Pérez-Ortiz et al. [2020]). The PBB training methods presented here are performant yet simple and elegant: These promising results are achieved i) with prior distributions 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, thus in line with the usual PAC-Bayes priors) and ii) via classical SGD optimization of a PAC-Bayes inspired training objective.

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

NeurIPS 2020 Workshop - Beyond BackPropagation: Novel Ideas for Training Neural Architectures.

2 Generalization and PAC-Bayes bounds

The ultimate goal for predictive machine learning algorithms is to find a solution that generalizes well, meaning intuitively that the learned model should give good predictions on unseen data. More formally, a training algorithm receives a size-n random sample $S = (Z_1, \ldots, Z_n)$, where $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} \subset \mathbb{R}$ and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ for supervised learning, and aims to find $w \in \mathcal{W}$ that minimises the empirical risk $\hat{L}_S(w) = n^{-1} \sum_{i=1}^n \ell(w, Z_i)$, or some regularized form of it, where $\mathcal{W} \subset \mathbb{R}^p$ and $\ell : \mathcal{W} \times \mathcal{Z} \to [0, \infty)$ is a fixed loss function, e.g. the commonly used cross-entropy loss for classification, which is used as a surrogate loss in lieu of the zero-one loss. More details on the learning framework are given in Section 5.1. See also Pérez-Ortiz et al. [2020].

While the outcome of training a classical neural network is a data-dependent weight vector, the outcome of training a probabilistic neural network is a data-dependent distribution over weights Q_S. Then, given a fresh input X, the network predicts its label by drawing a weight vector W at random from Q_S and applying predictor h_W to X. To measure the performance of the resulting randomizing predictor we use the expected loss over the random draws of weights. The average empirical loss becomes $Q_S[\hat{L}_S] = \int_{\mathcal{W}} \hat{L}_S(w) Q_S(dw)$ and the average loss becomes $Q_S[L] = \int_{\mathcal{W}} L(w) Q_S(dw)$.

The PAC-Bayes-kl theorem (Langford and Seeger [2001], Seeger [2002], Maurer [2004]) concludes that, as long as the loss function ℓ is bounded in $[0, 1]$, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over random n-samples S, simultaneously for all distributions Q over $\mathcal{W}$ it holds that:

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

where $\mathrm{KL}(Q\|P)$ is the Kullback-Leibler divergence from Q to P, and in the left-hand side $\mathrm{kl}(q\|p)$ is the binary KL divergence (i.e. the divergence from the Bernoulli distribution with parameter q to the Bernoulli distribution with parameter p), and Q^0 is a data-free distribution over $\mathcal{W}$, meaning 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 using Pinsker's inequality $\mathrm{kl}(\hat{p}\|p) \geq 2(p - \hat{p})^2$ and solve the resulting inequality for $Q[L]$ (see e.g. Tolstikhin and Seldin [2013]). Alternatively, one can use the refined version of Pinsker's inequality $\mathrm{kl}(\hat{p}\|p) \geq (p-\hat{p})^2/(2p)$ which is valid for $\hat{p} < p$ (see e.g. [Boucheron et al., 2013, Lemma 8.4]), the latter being tighter than the former when $p < 1/4$ (see details in Pérez-Ortiz et al. [2020]), and thus get:

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

However, due to the appearance of $Q[L]$ on the right-hand side this bound is not immediately useful for optimization purposes. One can view the above inequality as a quadratic inequality on $\sqrt{Q[L]}$. Solving this inequality for $Q[L]$ leads to the PAC-Bayes-quadratic bound shown below in Eq. (2). Alternatively, using ($\star$) combined with the inequality $\sqrt{ab} \leq \frac{1}{2}(\lambda a + \frac{b}{\lambda})$ valid for all $\lambda > 0$ and after some derivations, leads to the PAC-Bayes-λ bound of Thiemann et al. [2017] shown below in Eq. (3). For convenience, we quote the classical PAC-Bayes bound of McAllester [1999] in Eq. (4) below. The original proof of McAllester [1999] gave a slightly looser bound. The form presented in Eq. (4) is with the sharp dependence on n due to Maurer [2004]. The three bounds are:

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

$$Q[L] \leq \frac{Q[\hat{L}_S]}{1 - \lambda/2} + \frac{\mathrm{KL}(Q\|Q^0) + \log(2\sqrt{n}/\delta)}{n\lambda(1 - \lambda/2)}\,. \tag{3}$$

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

These inequalities are valid for losses with range [0,1]. Each one gives an upper bound on $Q[L]$ that holds with high probability (over the random draw of size-n samples) simultaneously for all distributions Q over weights. 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.

3 PAC-Bayes with Backprop

The essential idea of "PAC-Bayes with Backprop" is to train probabilistic neural networks (realized as distributions over the weight space) by minimizing a PAC-Bayes upper bound on the risk. Here we present two new training objectives, derived from Eq. (2) and Eq. (3) respectively, in the context of multiclass neural network classification. The standard surrogate loss used on these 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 $z \in \mathbb{R}^k$, $y \in [k] = \{1, \ldots, k\}$ and $\sigma : \mathbb{R}^k \to [0, 1]^k$ is the soft-max function, $\sigma(z)_i = \exp(z_i)/\sum_j \exp(z_j)$. Since the PAC-Bayes bounds of Eq. (2), (3) and (4) require losses within [0,1], we enforce an upper bound on the cross-entropy loss by lower-bounding the network probabilities by a value $p_{\min} > 0$ (cf. Dziugaite and Roy, 2018), which effectively puts an upper-bound $\ell_{\max}$ on the cross-entropy loss, and then normalizing by $\ell_{\max}$ gives a loss $\tilde{\ell}_1^{\text{x-e}}$ with range [0,1]. We thus propose to replace the zero-one loss with $\tilde{\ell}_1^{\text{x-e}}$ in either of Eq. (2), (3) or (4), leading to the three objectives:

$$f_{\text{quad}}(Q) = \left(\sqrt{Q[\hat{L}_S^{\text{x-e}}] + \frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} + \sqrt{\frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \right)^2 \tag{5}$$

$$f_{\text{lambda}}(Q, \lambda) = \frac{Q[\hat{L}_S^{\text{x-e}}]}{1 - \lambda/2} + \frac{\text{KL}(Q\|Q^0) + \log(2\sqrt{n}/\delta)}{n\lambda(1 - \lambda/2)} \tag{6}$$

$$f_{\text{classic}}(Q) = Q[\hat{L}_S^{\text{x-e}}] + \sqrt{\frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \tag{7}$$

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

Optimization of the objectives (5) and (7) entails minimizing over Q only, while optimization of (6) is done by alternating minimization with respect to Q and λ, similar to the procedure that was used by Thiemann et al. [2017] for training SVMs. By choosing Q in a parametric family of distributions, 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 $V \in \mathbb{R}^{p'}$ 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 $v = V$.[2] 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 optimization uses $\sigma = \log(1 + \exp(\rho))$, thus gradient updates are with respect to μ and ρ.

3.1 Prior distributions

We experiment with i) priors centered at randomly initialised weights and ii) priors centered at weights 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]. We test both Gaussian and Laplace prior distributions.

[2] Indeed, $\frac{\partial}{\partial\theta}\int Q_\theta(dw)\ell(h_w(x), y) = \frac{\partial}{\partial\theta}\int P_V(dv)\ell(h_{f_\theta(v)}(x), y) = \int P_V(dv)\frac{\partial}{\partial\theta}\ell(h_{f_\theta(v)}(x), y)$, 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. See e.g. Ruiz et al. [2016].

3.2 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 in Eq. (1), which is tighter than Eq. (2), (3) or (4), as the latter are relaxations of Eq. (1). To do so, we need to invert the binary KL from Eq. (1) and use Monte Carlo sampling to estimate the empirical term $Q[\hat{L}_S]$ by random weight sampling. We elaborate further on this in the appendix (Section 5.2).

4 Results and conclusions

We experimented on MNIST and CIFAR-10 to investigate the properties of the training objectives presented before. We evaluate the proposed f_{quad} and f_{lambda} of Eq. (5) and Eq. (6), and compare these to f_{classic} of Eq. (7) and the objective from Blundell et al. [2015] (f_{bbb}). We also compare to empirical risk minimisation with dropout (f_{erm}). The code for our experiments is publicly available[3]. All risk certificates were computed using the the PAC-Bayes-kl theorem, as explained in the appendix, with $\delta = 0.025$ and $\delta' = 0.01$ and $m = 150.000$ Monte Carlo model samples. We performed a grid search over all hyper-parameters and selected the run with the best risk certificate. We tested in our model selection experiments that the risk certificate has a strong correlation with test set error (Pearson correlation of 0.977), thus we use the risk and avoid the need for data splitting protocols (except for f_{erm}). Further details on the experiments can be found in Section 5.3.

The results are shown in percentages in Table 1. For all probabilistic neural networks we show the test set error of the stochastic classifier, while for f_{erm} we use classical neural networks and thus a deterministic classifier (included for comparison purposes). Several conclusions can be drawn from these results: Firstly, we see that a randomly initialised prior leads to worse minima than learning the prior on a subset of the data. Specifically, learnt priors lead to tight risk certificates (specially for PAC-Bayes inspired training objectives). This is consistent across the two tested architectures in MNIST. Thirdly, the test set errors coming from these new training objectives with learnt priors are close to the test error of f_{erm} and f_{bbb}. Finally, the risk certificates for MNIST are tighter than those for CIFAR-10, but these are still non-vacuous and relatively tight. For a more in depth analysis of these experiments we refer the reader to the appendix (Section 5.4).

Dataset, architecture and prior	f_{quad}		f_{lambda}		f_{classic}		f_{bbb}		f_{erm}
	Test err.	Risk ℓ^{01}	Test err.	Risk ℓ^{01}	Test err.	Risk ℓ^{01}	Test err.	Risk ℓ^{01}	Test err.
MNIST, FCN, rand Gauss	9.21	31.55	7.32	32.75	14.11	33.04	2.93	55.16	1.52
MNIST, FCN, learnt Gauss	2.02	2.79	1.96	3.54	2.30	2.84	1.79	9.68	1.52
MNIST, CNN, rand Gauss	5.35	21.65	3.97	22.02	8.69	22.77	1.54	36.45	0.92
MNIST, CNN, learnt Gauss	1.04	1.55	1.06	1.86	1.23	1.54	1.04	5.38	0.92
MNIST, CNN, rand Laplace	6.77	24.25	4.61	25.40	2.96	24.89	1.39	44.87	0.92
MNIST, CNN, learnt Laplace	1.39	1.67	0.98	2.16	1.07	1.55	0.92	8.66	0.92
CIFAR10, CNN (13 lay.) learnt Gauss	15.68	18.32	15.41	21.77	15.88	17.58	15.08	58.58	15.66
CIFAR10, CNN (15 lay.) learnt Gauss	14.63	18.06	14.37	21.21	14.75	16.67	14.13	55.72	14.13

Table 1: Test 0-1 error and risk certificate on MNIST and CIFAR-10 for several architectures/priors.

The risk certificates in Table 1 are much tighter than the ones reported in the literature for MNIST with the same FCN architecture. Specifically, the best two models in Dziugaite and Roy [2018] showed respectively a 12.00% test set error and a risk of 21.00%, and a 6% test set error and a risk certificate of 65%. The best two models in Lever et al. [2013] showed respectively a 12.00% test set error and a risk of 26.00%, and a 6% test set error and a risk certificate of 100%. In comparison we report a 2.00% test set error and 2.79% risk certificate for the same architecture and dataset.

The three tested PAC-Bayes inspired training objectives show some consistent differences. Specifically, f_{lambda} usually leads to more accurate predictors in test set error but looser risk certificates. f_{quad} seems to achieve better test set error and risk certificates for MNIST, while f_{classic} achieves tighter bounds for CIFAR-10. In any case, the results reported suggest that this family of methods shows promise towards self-certified learning, in the sense of certifying the risk on any unseen data without the need for data-splitting protocols (train and test and model selection).

[3]PyTorch code available at https://github.com/mperezortiz/PBB.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *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.

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.

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

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

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.

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

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.

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.

María Pérez-Ortiz, Omar Rivasplata, John Shawe-Taylor, and Csaba Szepesvári. Tighter risk certificates for neural networks. arXiv:2007.12911, 2020.

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

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.

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.

5 Appendices

5.1 Generalization via risk upper bounds

The ultimate goal for predictive machine learning algorithms is to find a solution that generalizes well, meaning intuitively that the learned model should give good predictions on unseen data. More formally, a training algorithm receives a size-n random sample $S = (Z_1, \ldots, Z_n)$, where $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} \subset \mathbb{R}$ and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ for supervised learning, and aims to find $w \in \mathcal{W}$ that minimises the empirical risk $\hat{L}_S(w) = n^{-1} \sum_{i=1}^n \ell(w, Z_i)$, or some regularized form of it, where $\mathcal{W} \subset \mathbb{R}^p$ and $\ell : \mathcal{W} \times \mathcal{Z} \to [0, \infty)$ is a fixed loss function, e.g. the commonly used cross-entropy loss for classification, which is used as a surrogate loss in lieu of the zero-one loss. An ideal goal would be to minimize the expected loss on unseen examples, also called the risk $L(w) = \mathbb{E}[\ell(w, Z)]$; however, the latter objective is inaccessible in any practical situation of interest, given that the data-generating distribution is unknown. Hence, the objective used in practice is the (possibly regularised) empirical risk. However, often minimizing the empirical risk can lead to situations in which the risk of the learned weight is significantly larger than the empirical risk —a case of overfitting. An alternative to the empirical risk is to choose a training objective which is guaranteed to give an upper bound on the risk, i.e. a so-called risk bound. Then, as long as the risk upper bound is tight and the optimization gives rise to a small value for this objective, the risk will also be small. By doing this, overfitting can be prevented, while we automatically get a self-certified learning method (cf. Freund [1998], Langford and Blum [2003]). Our paper Pérez-Ortiz et al. [2020] shows that this last approach is feasible with training objectives derived from PAC-Bayes bounds.

5.2 Computing the risk certificates

We compute a risk certificate on the error of the stochastic predictor following the procedure of Langford and Caruana [2001], which uses the PAC-Bayes-kl theorem. We describe now how to invert the binary KL from Eq. (1). 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. The crucial property that we rely on is that $\mathrm{kl}(x\|y) \leq b$ holds precisely when $y \leq f^\star(x, b)$.

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:

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

The difficulty is evaluating $Q[\hat{L}_S]$ since it 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 $Q[\hat{L}_S]$.

We also use $f^\star$ 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 $Q[\hat{L}^{01}_S]$ by setting the underlying loss function to be the 01 (classification) loss. This estimation is valid with high probability (of at least $1 - \delta'$) over random weight samples.

We evaluate the risk certificates (risk upper bounds) for the 0-1 loss 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 $W_1, \ldots, W_m \sim Q_S$ 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 this upper bound to evaluate risk certificates for the stochastic predictor.

5.3 Experiment details

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[4]. We did a grid sweep over the prior distribution scale hyper-parameter (i.e. standard deviation σ_0) with values in $[0.1, 0.05, 0.04, 0.03, 0.02, 0.01, 0.005]$. For the SGD with momentum optimizer we performed a grid sweep over learning rate in $[1\mathrm{e}-3, 5\mathrm{e}-3, 1\mathrm{e}-2]$ and momentum in $[0.95, 0.99]$. We 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. We observed that the value $p_{\min} = 1\mathrm{e}-5$ performed well. 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 $[1\mathrm{e}-5, 1\mathrm{e}-4, \ldots, 1\mathrm{e}-1]$, 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.

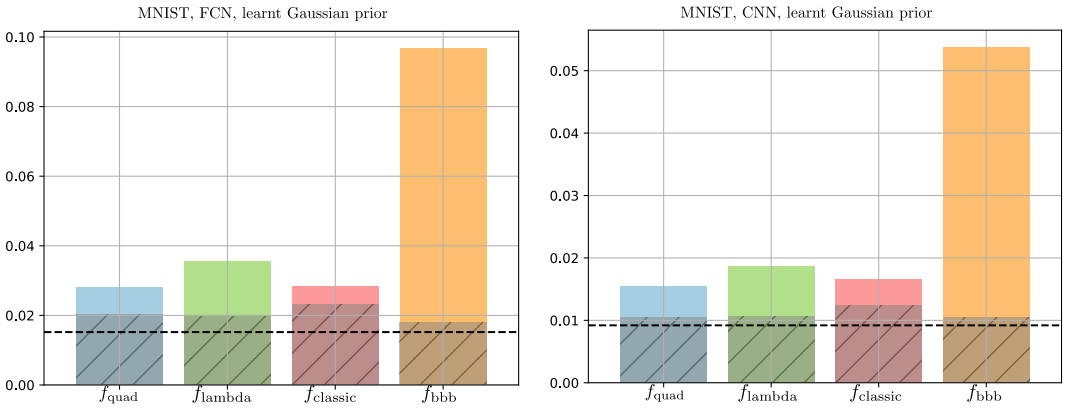

Figure 1: Bar plots of results across two architectures and training objectives for MNIST. The bottom shaded areas correspond to the test set 0-1 error of the stochastic classifier. The coloured areas on top correspond to the risk certificate. The horizontal dashed line corresponds to the test set 01 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).

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 two convolutional architectures (with 13 and 15 layers of learnable parameters) 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 earlier). We used a training batch size of 250 for all the experiments.

[4]Note that if we use a total of C hyperparameter combinations, the union bound correction would add no more than $\log(C)/30000$ to the PAC-Bayes-kl upper bound. Even with say $C = 42\mathrm{M}$ (forty two million) combinations, the value of our risk certificates will not be impacted significantly.

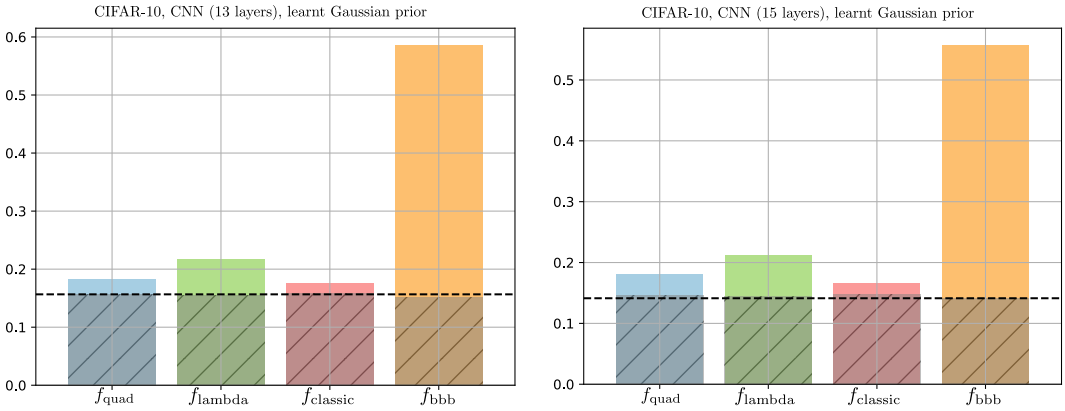

Figure 2: Bar plots of results achieved on CIFAR-10 for 2 different network architectures and all training objectives.

Priors We find that the prior learnt by ERM can be over-fitted easily. To avoid this, we use dropout during the learning process (exclusive to learning the prior, not the posterior). 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. 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. For MNIST we use 50% of the data for learning the prior, 100% of the data for refining the posterior and the remaining 50% for certifying the risk. For CIFAR-10 we use 70% of the data for learning the prior, 100% for refining the posterior and the remaining 30% for the risk.

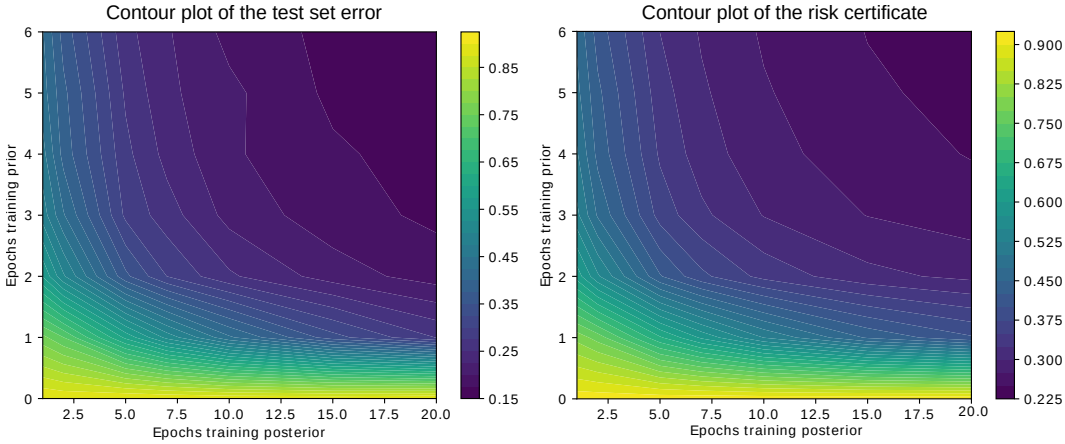

Figure 3: Contour plots of the test set error (0-1 loss) and risk certificate after different training epochs learning the prior and posterior and initial scale hyper-parameters value $\sigma_0 = 0.1$ for the prior. These experiments are performed on MNIST using the FCN architecture.

5.4 Additional results and analysis

The findings with learnt Gaussian priors from the experiments on MNIST and CIFAR-10 can be better seen in the bar plots in Figure 1 and Figure 2. These results show the following: i) f_{quad} achieves the best empirical performance and risk certificates for 0-1 error on MNIST, whereas all methods achieve similar test set performance for CIFAR-10 and f_{classic} obtains the tighter risk certificates. ii) f_{lambda} is the PAC-Bayes inspired objective that shows the looser risk certificates. iii) The results achieved by the stochastic predictor are close to those of empirical risk minimisation and f_{bbb}, while also providing tight risk certificates. This is obtained without the need for data splitting protocols. iv) 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 (and also deeper architectures) bring are also noteworthy.

Figure 3 shows a contour plot of the loss and risk certificate when training the prior and the posterior for different epochs (e.g. to check the effect of training the posterior with an under-fitted prior). This plot has been generated using the FCN architecture on MNIST with Gaussian priors. Similar results are obtained for the CNN architecture. Note that for the sake of visualisation we are plotting much less epochs that those used to generate the final results (in this case up to 20, whereas the previously reported results were with 100 epochs) so the reported test set errors and risk certificates differ from those previously reported. We observe that both training the prior and the posterior are crucial to improve the final loss and risk certificates, as the best loss and risk certificate values are found in the top right corner of the plot. The plot also shows that if the prior is under-fitted (e.g. if trained for only one or two epochs), then the final predictor can still be much improved with more training epochs for the posterior. However, a more adequate prior means that less epochs are needed to reach a reasonable posterior. Nonetheless, this is less apparent if the prior is not learnt (i.e. random, represented as a training of 0 epochs). In this experiment, we also noted that only a few epochs of training the prior are enough to reach competitive posteriors and that learning the posterior for much longer (e.g. 1000 epochs) does not lead to overfitting, which reinforces the finding of Blundell et al. [2015] that the KL term act as a regulariser. This is, however, opposed to what we observe when training the prior through empirical risk minimisation, since the prior overfits easily, which is why we had to learn the priors using dropout.

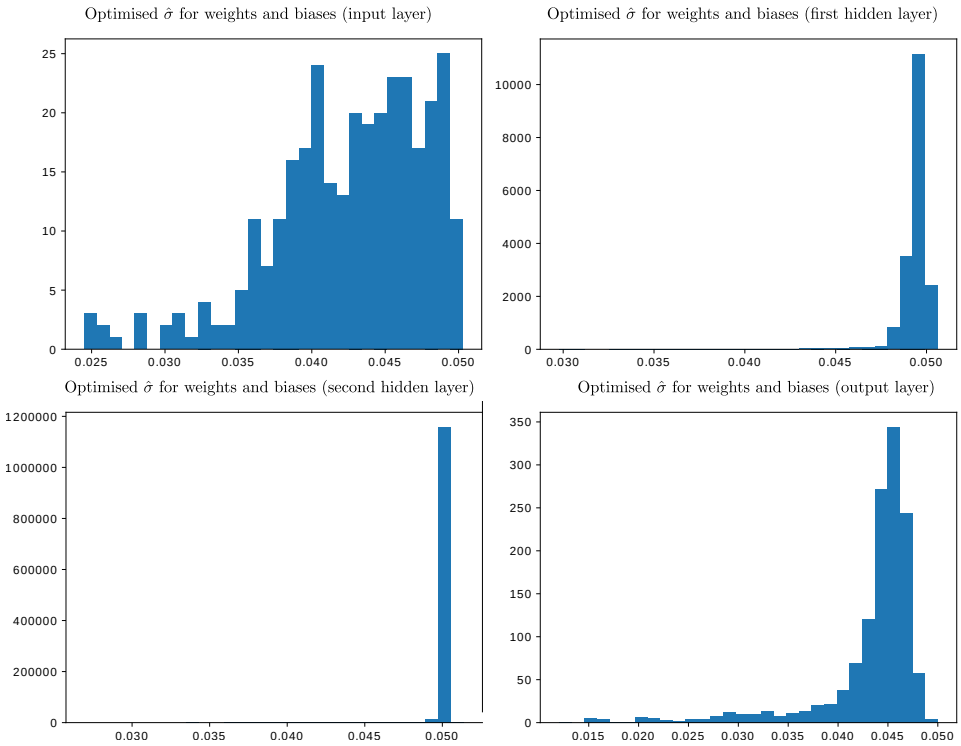

Figure 4: Histograms of the scale parameters for the Gaussian distribution at the end of the optimisation for the different layers of the CNN architecture on MNIST. All scale parameters were initialised to 0.05, i.e. $\sigma_0 = 0.05$ for all coordinates, and $\hat{\sigma}$ is the scale of the final output of training.

Figure 4 shows a histogram of the final scale parameters $\hat{\sigma}$ (i.e. standard deviation) for the Gaussian posterior distribution (both weights and biases). The plot shows that the optimisation changes the scale of different weights and biases, reducing specially those associated to the input and output layer. We will experiment with different scale initialisations per layer in our future work, as well as different covariance structures for the weight and bias distributions.