Wide stochastic networks: Gaussian limit and PAC-Bayesian training

Eugenio Clerico CLERICO@STATS.OX.AC.UK
George Deligiannidis DELIGIAN@STATS.OX.AC.UK
Arnaud Doucet DOUCET@STATS.OX.AC.UK
Department of Statistics, University of Oxford

Editors: Shipra Agrawal and Francesco Orabona

Abstract

The limit of infinite width allows for substantial simplifications in the analytical study of overparameterised neural networks. With a suitable random initialisation, an extremely large network exhibits an approximately Gaussian behaviour. In the present work, we establish a similar result for a simple stochastic architecture whose parameters are random variables, holding both before and during training. The explicit evaluation of the output distribution allows for a PAC-Bayesian training procedure that directly optimises the generalisation bound. For a large but finite-width network, we show empirically on MNIST that this training approach can outperform standard PAC-Bayesian methods.

Keywords: Infinite width; Gaussian limit; PAC-Bayes; Stochastic networks.

1. Introduction

In recent years, overparameterised artificial neural networks with millions of nodes have shown remarkably good generalisation capabilities. This behaviour contradicts the traditional well-rooted belief that overfitting is unavoidable when the trainable parameters far outnumber the size of the training dataset. It also highlights how the complexity bounds from classical statistical learning theory (Vapnik, 2000; Bousquet et al., 2004; Shalev-Shwartz and Ben-David, 2014) are manifestly inadequate tools to assess the generalisation properties of modern neural architectures (Zhang et al., 2021). As a consequence, the last couple of decades have seen the flourishing of novel results and techniques, aiming to explain the undeniable success of overparameterised models.

A large number of trainable parameters makes the direct study of a network's training dynamics extremely challenging. However, things become more manageable when approximations are made, as is the case in the limit of infinite width (Neal, 1995; Schoenholz et al., 2017; Yang, 2019; Hayou et al., 2019; Lee et al., 2019; Sirignano and Spiliopoulos, 2020; De Bortoli et al., 2020; Hayou et al., 2021). For a fully-connected feed-forward network, this limit consists in assuming that each layer includes an infinite number of nodes, while alternative definitions of *width* allow for extensions of this idea to encompass a vast range of architectures (Yang, 2019). Although unachievable in practice, infinitely wide networks feature the interesting property of behaving like Gaussian processes at initialisation, when all the parameters are suitably randomly initialised. This fact enable us to capture the output's behaviour of large (but finite-size) models, both before (Matthews et al., 2018; Lee et al., 2018) and during the training (Jacot et al., 2018).

In this work, we establish a similar asymptotic result for a simple stochastic architecture, featuring a single hidden layer. For a *stochastic* network, the randomness is not limited to the initialisation but is intrinsic in the parameters, which are treated as random variables. Specifically, here we assume that each parameter follows an independent normal distribution. As the architecture's width approaches infinity, we show that the network's output becomes Gaussian, with mean and covariance that can be derived from the means and standard deviations of the random parameters. We also show that under a lazy-regime assumption, where the parameters stay close to their initial values, this Gaussian behaviour is preserved throughout the training.

Part of the interest in studying stochastic networks is their role in the context of *learning with guarantees*, where the goal is to provide an upper-bound on the generalisation error without making use of any held-out test dataset. For long, in the overparameterised regime tight bounds could only be achieved under strong, and often hardly verifiable, hypotheses (Allen-Zhu et al., 2019). However, some promising non-vacuous results have been recently obtained by applying PAC-Bayesian methods to the training of stochastic classifiers (Dziugaite and Roy, 2017; Zhou et al., 2019; Pérez-Ortiz et al., 2021; Biggs and Guedj, 2022; Clerico et al., 2022).

The PAC-Bayesian theory originated from the seminal work of Shawe-Taylor and Williamson (1997), Shawe-Taylor et al. (1998), and McAllester (1998, 1999). We refer to Catoni (2007) for an extensive monograph on the topic, and to Guedj (2019) and Alquier (2021) for recent introductory overviews. It is a framework that provides upper bounds on the expected generalisation error of stochastic classifiers, with high probability over the random draw of the training dataset. The underlying idea is that if the distribution of the network's parameters does not change much during the training, then the learnt model should not be prone to overfit.

We call PAC-Bayesian training a procedure that aims to optimise a PAC-Bayesian bound. Often this optimisation cannot be tackled directly, as the distribution of the network's output is unknown, and one needs to sample multiple realisations of the stochastic parameters (Dziugaite and Roy, 2017; Pérez-Ortiz et al., 2021). In this paper, we propose to train a shallow wide stochastic network by exploiting the fact that it has an approximately Gaussian output. Notably, this approach allows for the direct optimisation of PAC-Bayesian bounds, even when a non-differentiable loss function, such as the 01-loss, is considered. We show empirically that the procedure that we present can bring tighter bounds and outperform standard PAC-Bayesian training methods.

As a final remark, it is worth mentioning that this is not the first work suggesting to exploit the output's Gaussianity to train a stochastic network. For instance, Alquier et al. (2016) uses a similar approach, but limited to a linear model for binary classification. Also, Clerico et al. (2022) built on a preprint of the current paper to develop a Gaussian training method for multilayer architectures.

2. Stochastic networks

We consider a simple network $\mathbb{R}^p \to \mathbb{R}^q$, consisting of a single hidden layer made of n nodes:

$$F(x) = W^1\,\phi(W^0 x)\,, \tag{1}$$

where W^1 is a $q \times n$ matrix, W^0 a $n \times p$ matrix, and ϕ the activation function. The network is stochastic. This means that W^0 and W^1 are random variables and each time a new input is fed to the network a new realisation of them is used to evaluate the output. Concretely, we let

$$W^1_{ij} = \tfrac{1}{\sqrt{n}}(\mathfrak{s}^1_{ij}\zeta^1_{ij} + \mathfrak{m}^1_{ij})\,; \qquad W^0_{jk} = \tfrac{1}{\sqrt{p}}(\mathfrak{s}^0_{jk}\zeta^0_{jk} + \mathfrak{m}^0_{jk})\,,$$

where $(\zeta^1_{ij})^{j=1\dots n}_{i=1\dots q}$ and $(\zeta^0_{jk})^{k=1\dots p}_{j=1\dots n}$ are independent families of iid standard normal random variables. We will henceforth call hyper-parameters the means $\mathfrak{m}$ and the standard deviations $\mathfrak{s}$, which are deterministic quantities when conditioned on their values at initialisation (possibly random).

We are interested in the infinite-width limit of large n. We aim at showing that, as $n \to \infty$, for each fixed input x the network's output $F(x)$ converges to a multivariate normal, whose covariance matrix $Q(x) \in \mathbb{R}^q \times \mathbb{R}^q$ and mean vector $M(x) \in \mathbb{R}^q$ are deterministic functions of the hyper-parameters $\mathfrak{m}$ and $\mathfrak{s}$. In short, for any fixed input x, we want to establish that

$$F(x) \to \mathcal{N}(M(x), Q(x))\,.^{1} \tag{2}$$

Note that, for two different inputs x and x', $F(x)$ and $F(x')$ are independent, as we assume that the stochastic parameters of the model are re-sampled every time that a new input is provided.

As a remark, by taking the limit $n \to \infty$ we mean considering a sequence of distinct networks of increasing widths, all initialised and trained in the same way. To be rigorous, one ought to add explicit superscripts ${}^{(n)}$ to the various quantities to stress their dependence on the network's width. So, one should actually write $F^{(n)}$, and say that its mean and covariance $M^{(n)}$ and $Q^{(n)}$ can be expressed in terms of $\mathfrak{m}^{(n)}$ and $\mathfrak{s}^{(n)}$. What we will show is that, for each x, $F^{(n)}(x) \to F(x) \sim \mathcal{N}(M(x), Q(x))$, where M and Q are the limits of $M^{(n)}$ and $Q^{(n)}$. However, we believe that stressing this explicit dependence on n would result in an excessively heavy notation. Therefore, we will always omit the superscript ${}^{(n)}$, and we will freely speak of "infinite-width limit" of a network, with the understanding that this has to be intended as the limit of a sequence of networks.

2.1. Infinite-width limit

We start by focusing on the hidden layer, which we denote as Y^0. Its nodes can be expressed as

$$Y_j^0(x) = \sum_{k=1}^{p} W_{jk}^0 x_k = \frac{1}{\sqrt{p}} \sum_{k=1}^{p} \mathfrak{s}_{jk}^0 \zeta_{jk}^0 x_k + \frac{1}{\sqrt{p}} \sum_{k=1}^{p} \mathfrak{m}_{jk}^0 x_k\,,$$

for any fixed input $x \in \mathbb{R}^p$. As the ζ_{jk}^0's are iid standard Gaussian random variables, we have that

$$Y^0(x) \sim \mathcal{N}(M^0(x), Q^0(x))\,.$$

This means that Y^0 is a n-dimensional multivariate normal, with mean vector and covariance matrix given by

$$M_j^0(x) = \frac{1}{\sqrt{p}} \sum_{k=1}^{p} \mathfrak{m}_{jk}^0 x_k\,; \qquad Q_{jj'}^0(x) = \delta_{jj'} \frac{1}{p} \sum_{k=1}^{p} (\mathfrak{s}_{jk}^0 x_k)^2\,.$$

As $Q^0(x)$ is diagonal, all the components of $Y^0(x)$ are independent, and we can actually write

$$Y_j^0(x) = \sqrt{Q_{jj}^0(x)}\,\bar{\zeta}_j^0 + M_j^0(x)\,, \tag{3}$$

where the $\bar{\zeta}_j^0$'s are independent standard normals.

Now, define the random variable

$$\Phi_j^0(x) = \phi(Y_j^0(x))\,.$$

Clearly, we have $F_i(x) = \sum_{j=1}^{n} W_{ij}^1 \Phi_j^0(x)$. Expanding the components of W^1 we can write

$$F_i(x) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \mathfrak{s}_{ij}^1 \zeta_{ij}^1 \Phi_j^0(x) + \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \mathfrak{m}_{ij}^1 \Phi_j^0(x)\,.$$

[1] Clearly, to be rigourous one needs to specify which kind of convergence is intended; see Propositions 3 and 4.

For any fixed input x, in the limit $n \to \infty$, we have an infinite sum of independent random variables, which are not identically distributed. In order to establish the convergence to a multivariate normal distribution, we need to control the variance and some higher moment of these variables, and hence require that the hyper-parameters have the correct order of magnitude. This is the case when the network is suitably initialised, and the result remains true during the training, as long as the hyper-parameters stay close enough to their initial values.

Note that for any finite width n, we can explicitly evaluate the network's mean M and covariance Q. For the mean, we have

$$M_i(x) = \mathbb{E}[F_i(x)] = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \mathfrak{m}^1_{ij} \mathbb{E}[\Phi^0_j(x)] \,. \tag{4}$$

As for $Q(x)$, we have $Q_{ii'}(x) = \mathbb{C}_{ii'}[F(x)] = \mathbb{E}[F_i(x)F_{i'}(x)] - \mathbb{E}[F_i(x)]\mathbb{E}[F_{i'}(x)]$, which becomes

$$Q_{ii'}(x) = \delta_{ii'} \frac{1}{n} \sum_{j=1}^{n} (\mathfrak{s}^1_{ij})^2 \mathbb{E}[\Phi^0_j(x)^2] + \frac{1}{n} \sum_{j=1}^{n} \mathfrak{m}^1_{ij} \mathfrak{m}^1_{i'j} \mathbb{V}[\Phi^0_j(x)] \,, \tag{5}$$

where we used the fact that the nodes of the hidden layer are independent and so the covariance of $\Phi^0(x)$ is diagonal. Once we will have established that the limit of infinite width leads to a Gaussian output, its mean and covariance will be given by the limit $n \to \infty$ of the above expressions.

We now state some rigorous results. The next proposition (see Appendix A for the proof) builds on a central limit theorem for triangular arrays, due to Bentkus (2005).

Proposition 1 *For any fixed input x and width n, define $M(x)$ and $Q(x)$ as in (4) and (5). Let $Z(x) \sim \mathcal{N}(M(x), Q(x))$ and denote as $\mathcal{C}$ the class of measurable convex subsets of $\mathbb{R}^q$. Let F be defined as in (1). Then*

$$\sup_{C \in \mathcal{C}} |\mathbb{P}(F(x) \in C) - \mathbb{P}(Z(x) \in C)| \leq \kappa q^{1/4} \frac{B(\mathfrak{m}, \mathfrak{s})}{\sqrt{n}} \,,$$

where $\kappa < 4$ is an absolute constant and

$$B(\mathfrak{m}, \mathfrak{s}) \leq q^{1/2} \frac{\frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{q} (2|\mathfrak{s}^1_{ij}|^3 + 8|\mathfrak{m}^1_{ij}|^3) \mathbb{E}[|\Phi^0_j(x)|^3]}{\left(\frac{1}{n} \sum_{j=1}^{n} \mathbb{E}[\Phi^0_j(x)^2] \min_{i=1\dots q} (\mathfrak{s}^1_{ij})^2 \right)^{3/2}} \,.$$

In particular, if $B(\mathfrak{m}, \mathfrak{s}) = o(\sqrt{n})$ for $n \to \infty$, then $F(x) - Z(x) \to 0$, in distribution.

As a corollary of the above result, if the stochastic network acts as a classifier, its performance is related to the one of its Gaussian approximation.

Corollary 2 *Assume that the network deals with a classification problem, where for each instance x there is a single correct label $y = f(x) \in \{1 \dots q\}$. With the notation of Proposition 1, for each fixed input $x \neq 0$, define as $\hat{f}(x) = \operatorname{argmax}_{i=1\dots q} F_i(x)$ and $\bar{f}(x) = \operatorname{argmax}_{i=1\dots q} Z_i(x)$. We have*

$$|\mathbb{P}(\hat{f}(x) = f(x)) - \mathbb{P}(\bar{f}(x) = f(x))| \leq \kappa q^{1/4} \frac{B(\mathfrak{m}, \mathfrak{s})}{\sqrt{n}} \,.$$

Proof For each $k = \{1 \dots q\}$, the set $\{z \in \mathbb{R}^q : z_k > \max_{i \neq k} z_i\}$ is convex. Hence the claim directly follows from Proposition 1. ■

2.2. Initialisation and lazy training

With a suitable random initialisation of the hyper-parameters, and in a lazy training regime, we show that, as $n \to \infty$, our stochastic network has a Gaussian limit, in the sense that the quantity $B/\sqrt{n}$ of Proposition 1 vanishes as $n \to \infty$. For simplicity, we shall assume that the activation function $\phi : \mathbb{R} \to \mathbb{R}$ is Lipshitz continuous (although we do not need to specify the Lipschitz constant).

We let all the network hyper-parameters be independently initialised in the following way:

$$\begin{aligned} \mathfrak{m}^0_{jk} &\sim \mathcal{N}(0,1)\,; \qquad & \mathfrak{m}^1_{ij} &\sim \mathcal{N}(0,1)\,; \\ \mathfrak{s}^0_{jk} &= 1\,; & \mathfrak{s}^1_{ij} &= 1\,, \end{aligned} \tag{6}$$

For convenience we write $\hat{\mathbb{P}}$ for the probability measure representing the above initialisation, while $\mathbb{P}$ is the probability measure describing the intrinsic stochasticity of the network. These two sources of randomness are always assumed to be independent.

Proposition 3 (Initialisation) *Consider a sequence of networks of increasing width initialised according to (6), and whose activation function ϕ is Lipshitz continuous. For any fixed input $x \neq 0$, defining B as in Proposition 1, we have $\frac{B(\mathfrak{m},\mathfrak{s})}{\sqrt{n}} \to 0$, as $n \to \infty$, in probability with respect to the random initialisation $\hat{\mathbb{P}}$. More precisely, $B(\mathfrak{m},\mathfrak{s}) = O(1)$ wrt $\hat{\mathbb{P}}$, as $n \to \infty$. In particular, at the initialisation the network tends to a Gaussian limit, in distribution wrt the intrinsic stochasticity $\mathbb{P}$ and in probability wrt $\hat{\mathbb{P}}$.*

Proof's sketch The proof is deferred to Appendix A. The main idea is that, since all the hyper-parameters are independent under (6), the standard central limit theorem yields that the upper-bound for B stated in Proposition 1 tends to a finite limit as $n \to \infty$. ■

The next proposition states that the limit will still be valid as long as the hyper-parameters do not move too much from their initialisation (lazy training).

Proposition 4 (Lazy training) *Fix a constant $J > 0$ independent of n, and assume that ϕ is Lipshitz. For a network of width n, with initial configuration $(\widetilde{\mathfrak{m}}, \widetilde{\mathfrak{s}})$ drawn according to $\hat{\mathbb{P}}$ as in (6), denote as $\mathcal{B}_J$ the ball*

$$\mathcal{B}_J = \left\{(\mathfrak{m},\mathfrak{s}) \;:\quad \|\mathfrak{m}^0 - \widetilde{\mathfrak{m}}^0\|^2_{F,2} + \|\mathfrak{m}^1 - \widetilde{\mathfrak{m}}^1\|^2_{F,2} + \|\mathfrak{s}^0 - \widetilde{\mathfrak{s}}^0\|^2_{F,2} + \|\mathfrak{s}^1 - \widetilde{\mathfrak{s}}^1\|^2_{F,2} \le J^2\right\}\,,$$

where $\|\cdot\|_{F,2}$ denotes the 2-Frobenius norm of a matrix. Let B be defined as in Proposition 1. For any fixed input $x \neq 0$ we have $B(\mathfrak{m},\mathfrak{s}) = O(1)$ as $n \to \infty$, uniformly on $\mathcal{B}_J$, in probability with respect to the random initialisation $\hat{\mathbb{P}}$.

Proof's sketch The proof is rather long and technical, and is deferred to Appendix A. However, the idea is simple and consists in showing that, under the lazy training assumption $(\mathfrak{m},\mathfrak{s}) \in \mathcal{B}_J$, B undergoes a change of order $O(1)$ during the training. Since by Proposition 3 we know that B is of order $O(1)$ at the initialisation, we can conclude. ■

In the next section, we will see that the lazy training constraint can be restated in terms of a bound on the Kullback-Leibler divergence between the initial and final distributions of the stochastic parameters. This fact will allow us to ensure that the constraint is satisfied when training the network to optimise a PAC-Bayesian objective.

3. PAC-Bayesian framework

Consider a standard classification problem, where to each instance $x \in \mathcal{X} \subseteq \mathbb{R}^p$ corresponds a unique correct label $y = f(x) \in \mathcal{Y} = \{1 \dots q\}$. The goal is to build an algorithm that is able to find a good prediction of y given x. We assume that the x's are distributed according to some probability measure $\mathbb{P}_X$ on $\mathcal{X}$. To train our algorithm, we have access to a sample $S = (X_h)_{h=1\dots m}$, which is correctly labelled (for every $X_h \in S$ we know $f(X_h)$). Each X_h is an independent draw from $\mathbb{P}_X$, so that $\mathbb{P}_S = \mathbb{P}_X^{\otimes m}$. We let ℓ be the 01-loss:

$$\ell(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y; \\ 1 & \text{otherwise.} \end{cases}$$

We let $\hat{f}_w(x)$ denote the prediction for the instance x, for a network with parameter configuration w. The empirical loss $L_S(w) = \frac{1}{m}\sum_{x\in S} \ell(\hat{f}_w(x), f(x))$ is the average of the 01-loss on the training set, while the true loss is $L_X(w) = \mathbb{E}_X[\ell(\hat{f}_w(X), f(X))]$.

The PAC-Bayesian framework (McAllester, 1998, 1999; Catoni, 2007; Guedj, 2019; Alquier, 2021) deals with stochastic neural classifiers. We consider a prior probability measure $\mathcal{P}$ on the random parameters, which has to be chosen independently of the specific realisation of the random dataset S used for the training. After the training, the parameters will be described by a new probability measure $\mathcal{Q}$ (the so-called posterior), clearly S-dependent. The idea is that if $\mathcal{P}$ and $\mathcal{Q}$ are not too “far” from each other, then the network will generalise well.

Under the posterior, we define the expected true loss $L_X(\mathcal{Q}) = \mathbb{E}_{W\sim\mathcal{Q}}[L_X(W)]$ and the expected empirical loss $L_S(\mathcal{Q}) = \mathbb{E}_{W\sim\mathcal{Q}}[L_S(W)]$. The PAC-Bayesian bounds are upper bounds on $L_X(\mathcal{Q})$, which hold with high probability on the random draw of the training set S. They usually involve the expected empirical error $L_S(\mathcal{Q})$ and a divergence term in the form of the Kullback-Leibler divergence between $\mathcal{Q}$ and $\mathcal{P}$: $\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) = \mathbb{E}_{\mathcal{Q}}[\log(\mathrm{d}\mathcal{P}/\mathrm{d}\mathcal{Q})]$. We will use the following result, due to Langford and Seeger (2001) and Maurer (2004).

Proposition 5 *Fix a data-independent prior $\mathcal{P}$. With probability higher than $1-\delta$ on the choice of the training set $S = (X_h)_{h=1\dots m}$*[2]*,*

$$L_X(\mathcal{Q}) \leq \mathrm{kl}^{-1}\left(L_S(\mathcal{Q}) \middle| \frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2\sqrt{m}}{\delta}}{m} \right), \tag{7}$$

for any posterior $\mathcal{Q}$. Here, we have defined $\mathrm{kl}^{-1}(u|c) = \sup\{v \in [0,1] : \mathrm{kl}(u\|v) \leq c\}$*, where* $\mathrm{kl}(u\|v) = u\log\frac{u}{v} + (1-u)\log\frac{1-u}{1-v}$.

We can hence devise the following training algorithm (McAllester, 1998):

- Fix $\delta \in (0,1)$ and a prior $\mathcal{P}$ for the network stochastic parameters;
- Collect a sample S of m iid datapoints;
- Compute the optimal posterior $\mathcal{Q}$ minimising (7);
- Implement a stochastic network characterised by the law $\mathcal{Q}$.

In practice, in essentially any realistic scenario the algorithm above cannot be implemented. Hence, one has to simplify the problem requiring that $\mathcal{P}$ and $\mathcal{Q}$ belong to some simple class of distributions.

[2] Here we assume that the training set S has size $m \geq 8$.

3.1. PAC-Bayesian training

Following the approach of Dziugaite and Roy (2017), we assume that both $\mathcal{P}$ and $\mathcal{Q}$ are multivariate normal distributions with diagonal covariance matrices. In other words, the random parameters of the network are independent normal random variables. For the posterior, $\mathfrak{m}$ and $\mathfrak{s}$ denote the N-dimensional vectors of the means and the standard deviations, while $\widetilde{\mathfrak{m}}$ and $\widetilde{\mathfrak{s}}$ refer to the prior. In short, $\mathcal{P} = \mathcal{N}(\widetilde{\mathfrak{m}}, \operatorname{diag}(\widetilde{\mathfrak{s}}^2))$ and $\mathcal{Q} = \mathcal{N}(\mathfrak{m}, \operatorname{diag}(\mathfrak{s}^2))$. In this Gaussian setting, $\mathrm{KL}(\mathcal{Q}\|\mathcal{P})$ takes a simple form:

$$\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) = \frac{1}{2}\left(\sum_\alpha \left(\frac{\mathfrak{s}_\alpha}{\widetilde{\mathfrak{s}}_\alpha}\right)^2 - N + \sum_\alpha \left(\frac{\mathfrak{m}_\alpha - \widetilde{\mathfrak{m}}_\alpha}{\widetilde{\mathfrak{s}}_\alpha}\right)^2 + 2\sum_\alpha \log\frac{\widetilde{\mathfrak{s}}_\alpha}{\mathfrak{s}_\alpha}\right)\,, \tag{8}$$

where the index α runs over all the components of the hyper-parameters.

Now, the most troublesome term in (7) is $L_S(\mathcal{Q})$, which in general cannot be computed explicitly. However, we can obtain a Monte Carlo (MC) estimate $\hat{L}_S(\mathcal{Q})$ of this quantity, by sampling a few realisations of the parameters from $\mathcal{Q}$.

Now, the idea is to perform a gradient descent (GD) optimisation on the PAC-Bayesian objective (Dziugaite and Roy, 2017; Pérez-Ortiz et al., 2021). Note that (8) is differentiable with respect to $\mathfrak{m}$ and $\mathfrak{s}$ (which are the trainable hyper-parameters of the posterior). However, $\hat{L}_S(\mathcal{Q})$ has a null gradient almost everywhere, as this is the case for $L_S(w)$ for each realisation w used in the estimate. The standard way to overcome this issue is to use a surrogate of the 01-loss for the training, such as some variant of the cross-entropy (Dziugaite and Roy, 2017; Pérez-Ortiz et al., 2021). Notably, although $\hat{L}_S(\mathcal{Q})$ has a null gradient, this is not the case for $L_S(\mathcal{Q})$ (see Section 4.1 and Figure 1 for more details). Hence, if we know exactly the output's distribution of the stochastic network, we might be able to use the 01-loss directly without the need of any surrogate. This is indeed the case for the Gaussian limit, as we will see in the next section. In a similar spirit, Alquier et al. (2016) studied the training of a linear binary classifier with Gaussian parameters.

It is worth mentioning that similar considerations hold when using an almost everywhere constant activation function to train a stochastic network. In this regard, Germain et al. (2009); Letarte et al. (2019); Biggs and Guedj (2021) developed an interesting variant of PAC-Bayesian training for binary classifiers with the sign activation function ($\phi = \mathrm{sign}$). In that setting, the simple form of the output of each layer allows for a more explicit expression of the distribution of the hidden nodes, which permits overcoming the fact that the binary activation function is non-differentiable.

4. PAC-Bayesian training in the Gaussian limit

Instead of doing the standard PAC-Bayesian training with a surrogate loss, we can train our wide stochastic network by assuming that its Gaussian approximation is exact. However, once completed the training, we will need to evaluate the final bound without such an assumption.

At the initialisation, for a network initialised according to $\hat{\mathbb{P}}$ as in (6), the Gaussian approximation is asymptotically exact for large n. Moreover, the following lemma ensures that controlling the KL divergence is enough to claim that the network is in the lazy training regime of Proposition 4. Hence, a wide stochastic network is asymptotically Gaussian throughout its PAC-Bayesian training.

Lemma 6 *Define the multivariate Gaussian distributions* $\mathcal{P} = \mathcal{N}(\widetilde{\mathfrak{m}}, \operatorname{diag}(\widetilde{\mathfrak{s}}^2)) = \mathcal{N}(\widetilde{\mathfrak{m}}, \mathrm{Id})$ *and* $\mathcal{Q} = \mathcal{N}(\mathfrak{m}, \operatorname{diag}(\mathfrak{s}^2))$ *for the parameters of a stochastic network. We have*

$$\|\mathfrak{m}^0 - \widetilde{\mathfrak{m}}^0\|_{F,2}^2 + \|\mathfrak{m}^1 - \widetilde{\mathfrak{m}}^1\|_{F,2}^2 + \|\mathfrak{s}^0 - \widetilde{\mathfrak{s}}^0\|_{F,2}^2 + \|\mathfrak{s}^1 - \widetilde{\mathfrak{s}}^1\|_{F,2}^2 \leq 2\mathrm{KL}(\mathcal{Q}\|\mathcal{P})\,.$$

Proof From (8), we conclude noticing that $u^2 - 1 - 2\log u \geq (u-1)^2$, for all $u > 0$. ■

The rest of this section is organised as follows. First, we show that it is possible to get a non-zero gradient from the expected 01-loss in the Gaussian limit. Then, we discuss how to evaluate the gradients of the output's mean and covariance with respect to the hyper-parameters. Finally, we deal with how to obtain a rigorous PAC-Bayesian bound after the training.

4.1. Training with the 01-loss

For $x \in \mathcal{X} \subseteq \mathbb{R}^p$, we want to find the correct $y = f(x)$ among q possible labels $i = 1, \dots, q$. We consider a Gaussian network with output $F(x) \sim \mathcal{N}(M(x), Q(x))$, whose random prediction is $\hat{f}(x) = \operatorname{argmax}_{i=1\dots q} F_i(x)$. Denoting as ℓ the 01-loss, it is natural to aim at minimising $\mathbb{E}[\ell(\hat{f}(x), f(x))]$ (where the expectation is over the stochastic parameters), since this quantity is actually equal to the probability of making a mistake for x: $\mathbb{P}(\hat{f}(x) \neq f(x))$. As we want to tackle the problem by performing gradient descent optimisation, if we assume that we are able to differentiate $M(x)$ and $Q(x)$ with respect to the network trainable hyper-parameters, all we need is to evaluate $\nabla_M \mathbb{E}[\ell(\hat{f}(x), y)]$ and $\nabla_Q \mathbb{E}[\ell(\hat{f}(x), y)]$.

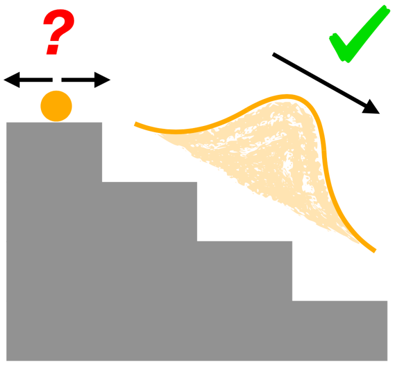

Figure 1: When "going down the stairs" via GD, each single realisation lies on a horizontal step and has an uninformative null gradient, but the whole distribution has a global view of the stairs and can find the good direction.

Note that $\ell(\hat{f}(x), y)$ has a null gradient almost everywhere for any random realisation of the network. For this reason, a non-stochastic network cannot be trained directly with the 01-loss. However, this is not the case for a stochastic network. The reason for that can be intuitively explained by thinking of what happens if we try to "go down the stairs" with GD, as illustrated in Figure 1. A single realisation of the network will be a point on a horizontal step: there is no way to understand the right direction in order to go down. However, if we consider the whole stochastic distribution of the network, it spreads over all the steps, and it has a global view of the stairs. It is hence not surprising that the gradient of the expected loss is non-zero.

For binary classification tasks, the expected 01-loss reads $\mathbb{E}[\ell(\hat{f}(x), 1)] = \mathbb{P}(F_2(x) > F_1(x))$ and $\mathbb{E}[\ell(F(x), 2)] = \mathbb{P}(F_1(x) > F_2(x))$. These quantities can be computed exactly:[3]

$$\mathbb{E}[\ell(\hat{f}(x), 1)] = \mathbb{P}_{\zeta\sim\mathcal{N}(0,1)}\left(\zeta > \frac{M_1(x) - M_2(x)}{\sqrt{Q_{11}(x) + Q_{22}(x) - 2Q_{12}(x)}}\right)\,;$$

$$\mathbb{E}[\ell(\hat{f}(x), 2)] = \mathbb{P}_{\zeta\sim\mathcal{N}(0,1)}\left(\zeta > \frac{M_2(x) - M_1(x)}{\sqrt{Q_{11}(x) + Q_{22}(x) - 2Q_{12}(x)}}\right)\,.$$

Clearly, the two expressions above can be written explicitly in terms of the error function erf, as ζ is distributed as a standard normal and $\mathbb{P}(\zeta > u) = \frac{1}{2}(1 - \operatorname{erf}(u/\sqrt{2}))$. It is then straightforward to see that $\mathbb{E}[\ell(\hat{f}(x), y)]$ is differentiable with respect to M and Q, with non-zero derivatives.

[3] A similar result was already derived in Alquier et al. (2016) for a simpler linear classifier.

When there are more than two classes, things become more complicated. It is however possible to exploit the Gaussianity and obtain a MC estimator of the expected loss, whose gradient with respect to M and Q is computable and not trivially zero. We refer to Appendix B for details.

4.2. Derivatives of M and Q

We have so far established that we can effectively differentiate the expected loss with respect to M and Q. Still, to train the network we will need to evaluate the gradients with respect to the hyper-parameters $\mathfrak{m}$ and $\mathfrak{s}$. Now, recall that $\Phi^0_k(x)$ is in the form $\phi(a\zeta + b)$, with $a = \sqrt{Q^0_{kk}(x)}$, $b = M^0_k(x)$, and $\zeta \sim \mathcal{N}(0,1)$. When the activation function ϕ is simple enough, $\mathbb{E}[\phi(a\zeta + b)]$ and $\mathbb{E}[\phi(a\zeta + b)^2]$ have closed-form expressions. Exploiting this fact, it is possible to evaluate the $\mathfrak{m}^0$- and $\mathfrak{s}^0$-derivatives of M and Q, needed in order to train the network with gradient-based methods. This if for instance the case for $\phi = \mathrm{ReLU}$ and $\phi = \sin$ (see Appendix C).

4.3. Final computation of the bound

Once completed the training, we need to abandon the Gaussian approximation to compute the final bound. We will follow the same approach as Dziugaite and Roy (2017) and Pérez-Ortiz et al. (2021).

Let $W_1, \dots, W_N$ be N independent realisations of the whole set of network stochastic parameters, drawn according to $\mathcal{Q}$. For $\delta' \in (0,1)$, with probability at least $1 - \delta'$ (Langford and Caruana, 2002)

$$L_S(\mathcal{Q}) \leq \mathrm{kl}^{-1}\left(\hat{L}_S(\mathcal{Q}) \middle| \tfrac{1}{N}\log\tfrac{2}{\delta'}\right) , \tag{9}$$

where kl^{-1} is defined in Proposition 5 and we have defined $\hat{L}_S(\mathcal{Q}) = \frac{1}{N}\sum_{h=1}^N L_S(W_h)$. Since kl^{-1} is increasing in its first argument, Proposition 5 yields that with probability at least $1 - \delta - \delta'$

$$L_X(\mathcal{Q}) \leq \mathrm{kl}^{-1}\left(\mathrm{kl}^{-1}\left(\hat{L}_S(\mathcal{Q}) \middle| \tfrac{1}{N}\log\tfrac{2}{\delta'}\right) \middle| \frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2\sqrt{m}}{\delta}}{m}\right) . \tag{10}$$

This method is often computationally very expensive, especially for large values of N. However, using a standard re-parameterisation trick from Kingma et al. (2015) helps to speed-up the evaluation, as it makes possible to obtain a realisation of the network by sampling only $d + n$ standard normals, instead of all the $p \times n^2 \times q$ stochastic parameters.

As a final remark, an alternative way to get an exact result from the Gaussian approximation is to use an upper bound, such as the one in Corollary 2, to control the finite-size correction to the expected empirical loss. However, for networks with $O(10^3)$ hidden nodes, like those that we used in our experiments, this last approach gives looser bounds compared to the method described above, at least when the number N of samples used for the MC estimate $\hat{L}_S(\mathcal{Q})$ is of order $O(10^5)$.

5. Experimental results

In this section, we present some empirical results to validate our theoretical findings. First, we compare the Gaussian predictions with the distribution of the output nodes of a wide stochastic network. Then, we report the results obtained by training a stochastic network on MNIST, and on a binary version of it, with our Gaussian method and with standard PAC-Bayesian procedures like those from Dziugaite and Roy (2017) and Pérez-Ortiz et al. (2021). On both datasets, the

Gaussian method led to tighter final generalisation bounds. The PyTorch code developped for this paper is available at https://github.com/eclerico/WideStochNet. For the sake of conciseness, we refer to Appendix D for an exhaustive account of the experimental details.

In order to keep the experimental setting as simple as possible, we opted for training only the means $\mathfrak{m}$ (keeping the standard deviations $\mathfrak{s}$ fixed at their initial value), similarly to what was done in Letarte et al. (2019). Moreover, coherently with the rest of this paper, all the networks that we used had no bias. The PAC-Bayesian priors were chosen in a completely data-independent fashion, and coincided with the distribution of the network at initialisation, as suggested by Dziugaite and Roy (2017).

We start by considering a toy dataset, whose datapoints were sampled from three multivariate standard normal distributions (labelled as 1, 2, 3) in $\mathbb{R}^4$, and then projected on the unit sphere in $\mathbb{R}^4$. A stochastic network with one hidden layer of $n = 1200$ nodes was trained to predict from which of the three Gaussian clusters each point comes. The histograms in Figure 2 represent the distributions of the network's output nodes, both before and after the training. They have been obtained for a single example by sampling 10^6 realisations of the random parameters. The theoretical predictions of the Gaussian profiles are plotted in black. The agreement with the histograms is striking, showing that the network is essentially Gaussian already for $O(10^3)$ hidden nodes.

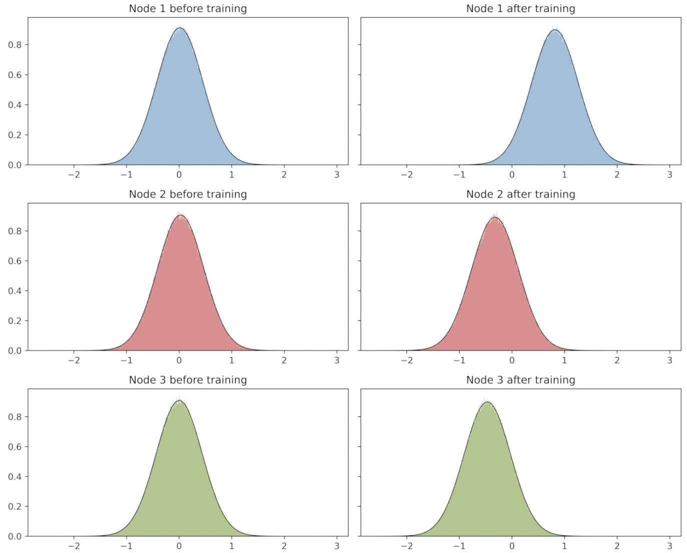

Figure 2: Distributions of the three output nodes of a wide stochastic network trained on a toy classification task. In black the theoretical predictions.

We now focus on the experiments on a binary version of the MNIST dataset, where the training dataset consisted of $m = 60000$ images. We considered a stochastic network with $n = 1200$ hidden nodes and ReLU activation function, initialised as in (6). We tried four training methods, based on different training objectives. The three "standard" PAC-Bayesian procedures used the objectives

$$\begin{aligned}
\texttt{McAll} &= \bar{L}_S(\mathcal{Q}) + \sqrt{\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2\sqrt{m}}{\delta}}{2m}}\,;\\
\texttt{lbd} &= \frac{\bar{L}_S(\mathcal{Q})}{(1-\lambda/2)} + \frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2\sqrt{m}}{\delta}}{m\lambda(1-\lambda/2)}\,;\\
\texttt{quad} &= \left(\sqrt{\bar{L}_S(\mathcal{Q}) + \frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2\sqrt{m}}{\delta}}{2m}} + \sqrt{\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2\sqrt{m}}{\delta}}{2m}}\right)^2,
\end{aligned} \tag{11}$$

where $\bar{L}_S(\mathcal{Q})$ is the expectation under $\mathcal{Q}$ of the empirical cross-entropy loss divided by $\log 2$. The objective McAll is from Dziugaite and Roy (2017), while quad comes from Pérez-Ortiz et al. (2021)

and `lbd` was originally derived by Thiemann et al. (2017) and later used by Pérez-Ortiz et al. (2021). In `lbd`, $\lambda \in (0,1)$ is also a trainable parameter.

As we are dealing with binary classification, for the "Gaussian" method (described Section 4), the expected value $L_S(\mathcal{Q})$ of the 01-loss can be evaluated directly (see Section 4.1). We could hence directly optimise (7), using the objective

$$\texttt{invkl} = \mathrm{kl}^{-1}\left(L_S(\mathcal{Q})\middle|\frac{\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2\sqrt{m}}{\delta}}{m}\right). \tag{12}$$

Table 1 illustrates the results of the experiment. The column "Bound" reports the values of the PAC-Bayesian bound (10). For the upper bound (9) on the empirical error, we used $N = 150000$ independent realisations of the net, $\delta' = 0.01$, and $\delta = 0.025$, so that the final generalisation bounds hold with probability higher than 0.965 on the random selection of the training set. The colum "Test Error" reports the average test error on a held-out dataset and its standard deviation. These values were evaluated on 10000 independent realisations of the test error. The two next columns refer to quantities computed within the Gaussian approximation: "G Bound" is the bound given by (7) and "G Loss" is the expected 01-loss. "Penalty" is the quantity $(\mathrm{KL}(\mathcal{Q}\|\mathcal{P}) + \log\frac{2\sqrt{m}}{\delta})/m$.

Table 1: Binary MNIST

Method	Bound	Test error	G Bound	G Loss	Penalty
`invkl`	**.1773**	$.0694_{\pm.0040}$	.1741	.0676	.0492
`McAll`	.1978	$\mathbf{.0456}_{\pm.0025}$	.1947	.0428	.1006
`lbd`	.1856	$.0543_{\pm.0030}$	.1825	.0520	.0752
`quad`	.1855	$.0533_{\pm.0030}$	.1823	.0515	.0757

Table 2: MNIST

Method	Bound	Test Error	G Bound	G Loss	Penalty
`invkl`	**.2807**	$\mathbf{.1083}_{\pm.0039}$	.2773	.1114	.0821
`McAll`	.4158	$.3189_{\pm.0097}$	.4120	.3265	.0155
`lbd`	.3736	$.2639_{\pm.0085}$	.3699	.2717	.0216
`quad`	.3735	$.2637_{\pm.0083}$	.3698	.2716	.0217

The "Gaussian" method yielded a tighter final bound than the "standard" ones. Yet, the best test error is achieved by `McAll`. It is worth noting that the final bound obtained with `McAll` is slightly worse than the one from Dziugaite and Roy (2017), where for a similar network of 1200 hidden nodes a final bound of .179 was obtained, whilst our result is .1921. However, our setting is simpler: our network has no bias, the standard deviations are not trained, and there is no choice of the optimal prior among different initialisations.

Finally, we report the results of a similar experiment on the full MNIST dataset (with the original 10 labels). The network is essentially the same one used for binary MNIST, with 1200 hidden nodes and ReLU activation function. The main difference is that now we have 10 output nodes. For the "standard" methods, we trained on the same objectives (11) as before, although this time we used a bounded version of the cross-entropy loss, as in Pérez-Ortiz et al. (2021). $\bar{L}_S(\mathcal{Q})$ is the expected value under $\mathcal{Q}$ of this bounded cross-entropy, averaged on the training set. The "Gaussian" method used the objective (12), where $L_S(\mathcal{Q})$ is again the expected empirical 01-loss. Actually, as we were dealing with more than two classes, we could not exactly compute the expected 01-loss, since we do not have a simple closed-form expression for it, and we proceeded as described in Appendix B.

Table 2 reports the results of the experiment on the full MNIST dataset, where for the estimate of the final bounds we again used $N = 150000$, $\delta' = 0.01$, and $\delta = 0.025$. Once more, the

"Gaussian" method obtained a tighter result, with almost a 0.1-gap with the bounds achieved by the other procedures. This time, the "Gaussian" method also attained the tightest test error. It is worth noticing that the PAC-Bayesian penalties of the standard methods are much lower than the respective losses[4], something that did not occur in Table 1. We conjecture that this behaviour is due to the different rescaling of the cross-entropy loss. On the other hand, this is not the case for the Gaussian method, as the loss does not require any rescaling.

6. Conclusions and perspectives

In the present work, we derive a Gaussian limit for a simple one-layer stochastic architecture, and point out how this result can be used in practice for the PAC-Bayesian training of wide shallow networks. First, we rigorously prove the validity of the limit at the initialisation and in a lazy training regime. Then, we show empirically that the proposed training method can outperform some standard PAC-Bayesian training procedures.

A main limitation of our approach is that it is limited to shallow networks with a single hidden layer. Indeed, our approach to establish the Gaussian limit relies on the fact that the hidden nodes are independent. This is not true anymore for any subsequent layer, and hence the CLT result that we use is no longer applicable. It is however worth mentioning that all the covariance matrices of the hidden layers are almost diagonal at the initialisation (as it is easy to check that the non-diagonal elements scale as $1/\sqrt{n}$) and a lazy-training constraint equivalent to the one in Proposition 4 might be enough to help establishing a rigorous Gaussian limit holding for multilayer architectures. In any case, even if one were able to use a limit theorem holding for the sum of weakly dependent nodes, evaluating the output's law of the network would require the knowledge of the (non-diagonal) covariance matrices of the hidden layers.[5] As we are looking at wide networks, the storage of these matrices would require a considerable amount of computational memory. Nevertheless, it is still possible to exploit our Gaussian PAC-Bayesian training ideas for multilayer architectures. This was recently done by Clerico et al. (2022), which built on our work to obtain PAC-Bayesian bounds using the fact that the network's output is Gaussian when conditioned on the hidden layers.

As a final remark, in the present work we did not treat the case of a network with biases. This is likely to be an elementary extension, which should not require much additional work.

Acknowledgments

EC is partially supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through the grant EP/R513295/1 (DTP scheme) and AD by EPSRC CoSInES EP/R034710/1. AD acknowledges support of the UK Defence Science and Technology Laboratory (DSTL) and EPSRC grant EP/R013616/1. This is part of the collaboration between US DOD, UK MOD and UK EPSRC under the Multidisciplinary University Research Initiative. The authors would like to thank Jian Qian for the valuable comments and suggestions.

[4] Training with longer time did not bring any relevant improvement, as the GD descent appeared to have already stabilised.

[5] Although the non diagonal elements are expected to scale as $1/\sqrt{n}$, the fact that they usually appear in sums of $O(n)$ terms can make their contribution non negligible. This was confirmed by a few empirical tests were we tried to only consider the diagonal elements of the covariance matrices of the central layers, and obtained inconsistencies between the predicted and the empirical output laws.

References

Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *NeurIPS*, 2019.

P. Alquier. User-friendly introduction to PAC-Bayes bounds. *arXiv:2110.11216*, 2021.

P. Alquier, J. Ridgway, and N. Chopin. On the properties of variational approximations of Gibbs posteriors. *Journal of Machine Learning Research*, 17, 2016.

V. Bentkus. A Lyapunov-type bound in Rd. *Theory of Probability & Its Applications*, 49(2), 2005.

F. Biggs and B. Guedj. Differentiable PAC-Bayes objectives with partially aggregated neural networks. *Entropy*, 23(10), 2021.

F. Biggs and B. Guedj. Non-vacuous generalisation bounds for shallow neural networks. *ICML*, 2022.

O. Bousquet, S. Boucheron, and G. Lugosi. *Introduction to Statistical Learning Theory*. Springer, 2004.

O. Catoni. PAC-Bayesian supervised classification: The thermodynamics of statistical learning. *IMS Lecture Notes Monograph Series*, 2007.

E. Clerico, G. Deligiannidis, and A. Doucet. Conditionally Gaussian PAC-Bayes. *AISTATS*, 2022.

V. De Bortoli, A. Durmus, X. Fontaine, and U. Simsekli. Quantitative propagation of chaos for SGD in wide neural networks. *NeurIPS*, 2020.

G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *UAI*, 2017.

P. Germain, A. Lacasse, F. Laviolette, and M. Marchand. PAC-Bayesian learning of linear classifiers. *ICML*, 2009.

B. Guedj. A primer on PAC-Bayesian learning. *Proceedings of the Second congress of the French Mathematical Society*, 2019.

S. Hayou, A. Doucet, and J. Rousseau. On the impact of the activation function on deep neural networks training. *ICML*, 2019.

S. Hayou, E. Clerico, B. He, G. Deligiannidis, A. Doucet, and J. Rousseau. Stable resnet. *AISTATS*, 2021.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *NeurIPS*, 2018.

D.P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. *NeurIPS*, 2015.

J. Langford and R. Caruana. (Not) bounding the true error. *NeurIPS*, 2002.

J. Langford and M. Seeger. Bounds for averaging classifiers. *CMU tech report*, 2001.

J. Lee, Y. Bahri, R. Novak, S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as Gaussian processes. *ICLR*, 2018.

J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 2019.

G. Letarte, P. Germain, B. Guedj, and F. Laviolette. Dichotomize and generalize: PAC-Bayesian binary activated deep neural networks. *NeurIPS*, 2019.

A. G. Matthews, J. Hron, M. Rowland, R.E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. *ICLR*, 2018.

A. Maurer. A note on the PAC Bayesian theorem. *arXiv:0411099*, 2004.

D.A. McAllester. Some PAC-Bayesian theorems. *COLT*, 1998.

D.A. McAllester. PAC-Bayesian model averaging. *COLT*, 1999.

R. M. Neal. Bayesian learning for neural networks. *Springer Science & Business Media*, 118, 1995.

M. Pérez-Ortiz, O. Risvaplata, J. Shawe-Taylor, and C. Szepesvári. Tighter risk certificates for neural networks. *Journal of Machine Learning Research*, 22, 2021.

S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein. Deep information propagation. *ICLR*, 2017.

S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.

J. Shawe-Taylor and R.C. Williamson. A PAC analysis of a Bayesian estimator. *COLT*, 1997.

J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5), 1998.

J. Sirignano and K. Spiliopoulos. Mean field analysis of neural networks: A law of large numbers. *SIAM Journal on Applied Mathematics*, 80(2), 2020.

N. Thiemann, C. Igel, O. Wintenberger, and Y. Seldin. A strongly quasiconvex PAC-Bayesian bound. *ALT*, 2017.

V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

G. Yang. Tensor programs I: Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *NeurIPS*, 2019.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3), 2021.

W. Zhou, V. Veitch, M. Austern, R.P. Adams, and P. Orbanz. Non-vacuous generalization bounds at the imagenet scale: a PAC-Bayesian compression approach. *ICLR*, 2019.

Appendix A. Omitted proofs

Throughout this section we use several notations for the norms of vectors and matrices. For $\gamma \geq 1$ and a vector v, $\|v\|_\gamma = (\sum_i |v_i|^\gamma)^{1/\gamma}$. If A is a matrix, we define $\|A\|_{F,\gamma} = (\sum_{ij} |A_{ij}|^\gamma)^{1/\gamma}$ and $\|A\|_\gamma = \sup_{v:\|v\|_\gamma=1} \|Av\|_\gamma$. We also recall that $\mathbb{P}$ denotes the intrinsic stochaticity of the network, while $\hat{\mathbb{P}}$ is the randomness due to the initialisation. These two sources of stochasticity are always supposed to be mutually independent. We denote as $\mathbb{E}$ the expectation wrt $\mathbb{P}$, and as $\hat{\mathbb{E}}$ the one wrt $\hat{\mathbb{P}}$. Moreover we write $\Gamma = O_{\hat{\mathbb{P}}}(n^\gamma)$ to mean that $\limsup_{n\to\infty} \frac{|\Gamma|}{n^\gamma} < \infty$ in probability wrt $\hat{\mathbb{P}}$, and $\Gamma = \Omega_{\hat{\mathbb{P}}}(n^\gamma)$ for $\limsup_{n\to\infty} \frac{|\Gamma|}{n^\gamma} > 0$ in probability wrt $\hat{\mathbb{P}}$.

We want to prove a rigorous result of convergence to the Gaussian limit of wide stochastic networks. We will essentially make use of the next result, due to Bentkus (2005).

Theorem 7 *Let $X_1, \dots, X_n$ be independent random vectors in $\mathbb{R}^q$, such that $\mathbb{E}[X_j] = 0$ for all j. Let $Y = \frac{1}{\sqrt{n}} \sum_{j=1}^n X_j$ and assume that the covariance matrix $\mathbb{C}[Y]$ is non singular. Let $Z \sim \mathcal{N}(0, \mathbb{C}[Y])$. Denote as $\frac{1}{\sqrt{\mathbb{C}[Y]}}$ the inverse of the positive square root of the matrix $\mathbb{C}[Y]$, and let $B_j = \mathbb{E}[\|\frac{1}{\sqrt{\mathbb{C}[Y]}} X_j\|_2^3]$ and $B = \frac{1}{n}\sum_{j=1}^n B_j$. Let $\mathcal{C}$ denote the class of all convex subsets of $\mathbb{R}^p$. Then, there exists an absolute positive constant $\kappa < 4$ such that*

$$\sup_{C\in\mathcal{C}} |\mathbb{P}(Y \in C) - \mathbb{P}(Z \in C)| \leq \kappa q^{1/4} \frac{B}{\sqrt{n}} \,. \tag{13}$$

Our goal is to prove a Gaussian limit as $n \to \infty$ for $F(x)$, whose components are given by

$$F_i(x) = \frac{1}{\sqrt{n}} \sum_{j=1}^n \mathfrak{s}_{ij}^1 \zeta_{ij}^1 \Phi_j^0(x) + \frac{1}{\sqrt{n}} \sum_{j=1}^n \mathfrak{m}_{ij}^1 \Phi_j^0(x) \,.$$

Let us denote by X_j the q-dimensional vector $X_j = (X_{1j} \dots X_{qj})$, with

$$X_{ij} = \mathfrak{s}_{ij}^1 \zeta_{ij}^1 \Phi_j^0(x) + \mathfrak{m}_{ij}^1 (\Phi_j^0(x) - \mathbb{E}[\Phi_j^0(x)]) \,.$$

Since all the ζ_{ij}^1's and the Φ_j^0's are independent, the X_j's constitute a family of n centred independent q-dimensional random vectors (wrt the intrinsic network stochasticity $\mathbb{P}$).

Clearly, we have $F(x) = \mathbb{E}[F(x)] + \frac{1}{\sqrt{n}} \sum_{j=1}^n X_j$. Let us define $Y = \frac{1}{\sqrt{n}} \sum_{j=1}^n X_j$. Note that, for all x, the covariance matrix $\mathbb{C}[Y]$ is given by $Q(x)$ (defined in (5)), no matter if Y is Gaussian or not. Using the same notations of Theorem 7, assuming that $\mathbb{C}[Y]$ is non-singular, we have

$$\sup_{C\in\mathcal{C}} |\mathbb{P}(Y \in C) - \mathbb{P}(Z \in C)| \leq \kappa q^{1/4} \frac{B}{\sqrt{n}} \,.$$

To prove that the Y behaves as a Gaussian for large n, we will show that $B = O(1)$ for large n. We can easily upperbound each B_j as

$$B_j \leq \frac{1}{\lambda[Y]^{3/2}} \mathbb{E}[\|X_j\|_2^3] \,,$$

where $\lambda[Y] > 0$ is the smallest eigenvalue of $\mathbb{C}[Y]$. For simplicity, we have omitted the dependence of these quantities on $\mathfrak{m}$ and $\mathfrak{s}$. We will often do so throughout this section, in order to lighten the notation.

Define $G = \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\|X_j\|_2^3]$ and $\Lambda = \frac{1}{n}\sum_{j=1}^{n}\lambda[X_j]$. Clearly, as the X_j's are independent, we have $\mathbb{C}[Y] = \frac{1}{n}\sum_{j=1}^{n}\mathbb{C}[X_j]$. In particular, we can easily find that $\lambda[Y] \geq \Lambda$.

We summarise what we have so far in the next lemma.

Lemma 8 *With the notations introduced above, assuming that $\mathbb{C}[Y]$ is not singular, we have*

$$B \leq \frac{1}{n}\sum_{j=1}^{n}\frac{1}{\Lambda^{3/2}}\mathbb{E}[\|X_j\|_2^3] = \frac{G}{\Lambda^{3/2}}\,.$$

Now, from Hölder's inequality we have $\|X_j\|_2 \leq \|\mathbf{1}_q\|_6\|X_j\|_3 = q^{1/6}\|X_j\|_3$, and so

$$\|X_j\|_2^3 \leq q^{1/2}\|X_j\|_3^3 = q^{1/2}\sum_{i=1}^{q}|X_{ij}|^3\,. \tag{14}$$

Then, with some simple algebraic manipulation, and applying Jensen's inequality, we obtain

$$\mathbb{E}[|X_{ij}|^3] \leq (|\mathfrak{s}_{ij}^1|^3\mathbb{E}[|\zeta_{ij}^1|^3] + 8|\mathfrak{m}_{ij}^1|^3)\mathbb{E}[|\Phi_j^0(x)|^3]\,.$$

For convenience, we introduce the following notations

$$H_{ij} = (2|\mathfrak{s}_{ij}^1|^3 + 8|\mathfrak{m}_{ij}^1|^3)\mathbb{E}[|\Phi_j^0(x)|^3]\,; \qquad H_j = \sum_{i=1}^{q}H_{ij}\,; \qquad H = \frac{1}{n}\sum_{j=1}^{n}H_j\,,$$

so that we have $G \leq q^{1/2}H$, since $\mathbb{E}[|\zeta|^3] = 2\sqrt{2/\pi} < 2$ for $\zeta \sim \mathcal{N}(0,1)$.

On the other hand, we can find a lowerbound for Λ as well. Indeed, first we can notice that

$$\mathbb{C}_{ii'}[X_j] = \delta_{ii'}(\mathfrak{s}_{ij}^1)^2\mathbb{E}[\Phi_j^0(x)^2] + \mathfrak{m}_{ij}^1\mathfrak{m}_{i'j}^1\mathbb{V}[\Phi_j^0(x)]\,.$$

The first term is a diagonal matrix, while the second one is non-negative definite. Hence we can write

$$\lambda[X_j] \geq \mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\ldots q}(\mathfrak{s}_{ij}^1)^2\,.$$

Defining

$$\Theta = \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\ldots q}(\mathfrak{s}_{ij}^1)^2 \leq \Lambda \tag{15}$$

we have the following corollary of Lemma 8.

Corollary 9 *With the same notations as above, if $\mathbb{C}[Y]$ is non singular, we have*

$$B \leq q^{1/2}\frac{H}{\Theta^{3/2}}\,.$$

Note that both H and Θ can be evaluated explicitly, given the parameters of the networks, as long as ϕ allows for an explicit evaluation of $\mathbb{E}[|\Phi^0(x)|^\gamma]$, for $\gamma = 1,2,3$. This means that we can give an exact upper bound to the finite-size error of the predicted 01-loss, for any configuration of the network.

We can now prove Proposition 1 and Corollary 2 from the main text.

Proposition 1 *For any fixed input x and width n, define $M(x)$ and $Q(x)$ as in (4) and (5). Let $Z(x) \sim \mathcal{N}(M(x), Q(x))$ and denote as $\mathcal{C}$ the class of measurable convex subsets of $\mathbb{R}^q$. Let F be defined as in (1). Then*

$$\sup_{C \in \mathcal{C}} |\mathbb{P}(F(x) \in C) - \mathbb{P}(Z(x) \in C)| \leq \kappa q^{1/4} \frac{B(\mathfrak{m}, \mathfrak{s})}{\sqrt{n}} \,,$$

where $\kappa < 4$ is an absolute constant and

$$B(\mathfrak{m}, \mathfrak{s}) \leq q^{1/2} \frac{\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{q}(2|\mathfrak{s}_{ij}^1|^3 + 8|\mathfrak{m}_{ij}^1|^3)\mathbb{E}[|\Phi_j^0(x)|^3]}{\left(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\dots q}(\mathfrak{s}_{ij}^1)^2\right)^{3/2}} \,.$$

In particular, if $B(\mathfrak{m}, \mathfrak{s}) = o(\sqrt{n})$ for $n \to \infty$, then $F(x) - Z(x) \to 0$, in distribution.

Proof The result is a straight consequence of Theorem 7 and Corollary 9. Note that $\mathbb{C}[Y]$ is non singular as long as all the components of $\mathfrak{s}$ are non-zero, so as long as the bound in the statement is finite. ■

In the next section we show how, with a suitable random initialisation, we can assure that the network has an almost Gaussian behaviour. Successively, we will show that this behaviour is preserved during training, as long as the hyper-parameters do not move too much from their initial values.

A.1. Initialisation

We consider the random initialisation:

$$\begin{aligned} \mathfrak{m}_{jk}^0 &\sim \mathcal{N}(0,1)\,; \qquad & \mathfrak{m}_{ij}^1 &\sim \mathcal{N}(0,1)\,; \\ \mathfrak{s}_{jk}^0 &= 1\,; & \mathfrak{s}_{ij}^1 &= 1\,, \end{aligned} \tag{6}$$

As now we have two sources of randomness (the initialisation and the intrinsic stochasticity of the network) to avoid confusion we will denote as $\hat{\mathbb{E}}$, $\hat{\mathbb{P}}$, $\hat{\mathbb{V}}$ the expectations, probabilities and variances with respect to the initialisation, whilst $\mathbb{E}$, $\mathbb{P}$ and $\mathbb{V}$ refer to the network intrinsic stochasticity.

Lemma 10 *Define H and Θ as in the previous section for a network with parameters $(\mathfrak{m}, \mathfrak{s})$ distributed according to $\hat{\mathbb{P}}$, as in (6). Assume that ϕ is Lipshitz continuous. Then, for any fixed $x \neq 0$, $H \to h > 0$ and $\Theta \to \theta > 0$ in probability as $n \to \infty$, with respect to the random initialisation, where both h and θ are finite.*

Proof First notice that, fixed an input $x \neq 0$ and fixed n, all the Φ_j^0's are iid, with respect to $\hat{\mathbb{P}}$, as all the components of $\mathfrak{m}^0$ and the $\mathfrak{s}_0$ are. As a consequence all the H_j's are iid with respect to $\hat{\mathbb{P}}$ (note that they have different distribution for different n as the law of the Φ_j^0's depends on n). Now, thanks to the fact that ϕ is Lipshitz continuous, we have that $\limsup_{n\to\infty} \hat{\mathbb{V}}[H_j] < \infty$. Hence, by a standard application of the CLT for triangular arrays, we get that

$$H - \hat{\mathbb{E}}[H] = \frac{1}{n}\sum_{j=1}^{n}(H_j - \hat{\mathbb{E}}[H_j]) \to 0$$

in distribution, and hence in probability, as 0 is a constant. It is quickly verified that the limit $h = \lim_{n\to\infty} \hat{\mathbb{E}}[H]$ exists, finite and positive. The proof for Θ is analogous. ■

Now we can easily prove Proposition 3.

Proposition 3 *Consider a sequence of networks of increasing width initialised according to (6), and whose activation function ϕ is Lipshitz continuous. For any fixed input $x \neq 0$, defining B as in Proposition 1, we have $\frac{B(\mathfrak{m},\mathfrak{s})}{\sqrt{n}} \to 0$, as $n \to \infty$, in probability with respect to the random initialisation $\hat{\mathbb{P}}$. More precisely, $B(\mathfrak{m},\mathfrak{s}) = O(1)$ wrt $\hat{\mathbb{P}}$, as $n \to \infty$. In particular, at the initialisation the network tends to a Gaussian limit, in distribution wrt the intrinsic stochasticity $\mathbb{P}$ and in probability wrt $\hat{\mathbb{P}}$.*

Proof It is a straight consequence of Lemma 10. ■

A.2. Lazy training

We have established that the Gaussian limit holds at initialisation. In the present section we will see that, as far as the hyper-parameters of the network do not move too much from their initial values, the limit keeps its validity.

Proposition 4 *Fix a constant $J > 0$ independent of n, and assume that ϕ is Lipshitz. For a network of width n, with initial configuration $(\widetilde{\mathfrak{m}}, \widetilde{\mathfrak{s}})$ drawn according to $\hat{\mathbb{P}}$ as in (6), denote as $\mathcal{B}_J$ the ball*

$$\mathcal{B}_J = \left\{(\mathfrak{m},\mathfrak{s})\ :\quad \|\mathfrak{m}^0 - \widetilde{\mathfrak{m}}^0\|_{F,2}^2 + \|\mathfrak{m}^1 - \widetilde{\mathfrak{m}}^1\|_{F,2}^2 + \|\mathfrak{s}^0 - \widetilde{\mathfrak{s}}^0\|_{F,2}^2 + \|\mathfrak{s}^1 - \widetilde{\mathfrak{s}}^1\|_{F,2}^2 \leq J^2\right\},$$

where $\|\cdot\|_{F,2}$ denotes the 2-Frobenius norm of a matrix. Let B be defined as in Proposition 1. For any fixed input $x \neq 0$ we have $B(\mathfrak{m},\mathfrak{s}) = O(1)$ as $n \to \infty$, uniformly on $\mathcal{B}_J$, in probability with respect to the random initialisation $\hat{\mathbb{P}}$.

Proof For convenience we will write with a tilde all the quantities relative to the network at initialisation. We denote with a Δ the difference between the final and the initial values of these quantities. For instance, $\Theta = \frac{1}{n}\sum_{j=1}^n \mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\dots q}(\mathfrak{s}_{ij}^1)^2$, $\widetilde{\Theta} = \frac{1}{n}\sum_{j=1}^n \mathbb{E}[\widetilde{\Phi}_j^0(x)^2]\min_{i=1\dots q}(\widetilde{\mathfrak{s}}_{ij}^1)^2$, and $\Delta\Theta = \Theta - \widetilde{\Theta}$.

We will show that for $n \to \infty$, $\Theta = \Omega_{\hat{\mathbb{P}}}(1)$ and $G = O_{\hat{\mathbb{P}}}(1)$ uniformly on $\mathcal{B}_J$, so that we can conclude using that $\Lambda \geq \Theta$ and Lemma 8.

Fix an input x. First, we need a bound on $\|\Delta\Phi^0(x)\|_2 = \|\Phi^0(x) - \widetilde{\Phi}^0(x)\|_2$. We have that $\Phi^0(x) = \phi(Y^0(x))$. Hence, letting L be the Lipshitz constant of ϕ, we have $\|\Delta\Phi^0(x)\|_2 \leq L\|\Delta Y^0(x)\|_2$. Now, as $\Delta Y_j^0(x) = \frac{1}{\sqrt{p}}\sum_{k=1}^p \Delta\mathfrak{m}_{jk}^0 x_k + \frac{1}{\sqrt{p}}\sum_{k=1}^p \Delta\mathfrak{s}_{jk}^0\zeta_{jk}^0 x_k$, we have

$$\|\Delta\Phi^0(x)\|_2 \leq \frac{L}{\sqrt{p}}(\|\Delta\mathfrak{m}^0\|_2 + \|\Delta\mathfrak{s}^0 \odot \zeta^0\|_2)\|x\|_2\,,$$

where $\odot$ denotes the Hadamard product.

Notice that we have

$$\mathbb{E}[\|\Delta\mathfrak{s}^0 \odot \zeta^0\|_2^2] \leq \mathbb{E}[\|\Delta\mathfrak{s}^0 \odot \zeta^0\|_{F,2}^2] = \sum_{j=1}^n\sum_{k=1}^p (\Delta\mathfrak{s}_{jk}^0)^2\mathbb{E}[(\zeta_{jk}^0)^2] = \|\Delta\mathfrak{s}^0\|_{F,2}^2 \leq J^2$$

uniformly in $\mathcal{B}_J$, where as usual the expectation $\mathbb{E}$ is the one with respect to the intrinsic stochasticity of the network, due to the ζ's. We can define a constant $C \geq 0$, independent of n, such that

$$\mathbb{E}[\|\Delta\Phi^0(x)\|_2^2] \leq \frac{4L^2J^2\|x\|_2^2}{p} = C^2$$

uniformly in $\mathcal{B}_J$, as $\|\Delta\mathfrak{m}^0\|_2 \leq \|\Delta\mathfrak{m}^0\|_{F,2} \leq J$.

Now, recalling the definition of Θ and using that $\mathfrak{s}^1 = 1 + \Delta\mathfrak{s}^1$, we have

$$\Theta = \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\dots q}(\mathfrak{s}_{ij}^1)^2 \geq \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2](1 - 2\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1|)\,.$$

We will show that $\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2] \to \widetilde{\Theta}$ and $\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1| \to 0$.

First notice that

$$\left|\frac{1}{n}\mathbb{E}[\|\Phi^0(x)\|_2^2] - \frac{1}{n}\mathbb{E}[\|\widetilde{\Phi}^0(x)\|_2^2]\right| \leq \frac{2}{n}\mathbb{E}[|\widetilde{\Phi}^0(x)\cdot\Delta\Phi^0(x)|] + \frac{1}{n}\mathbb{E}[\|\Delta\Phi_j^0(x)\|_2^2]\,.$$

We know that $\frac{1}{n}\mathbb{E}[\|\widetilde{\Phi}^0(x)\|_2^2] = \widetilde{\Theta}$ by definition. On the other hand we have

$$\begin{aligned}\frac{2}{n}\mathbb{E}[|\widetilde{\Phi}^0(x)\cdot\Delta\Phi^0(x)|] &\leq \frac{2}{n}\mathbb{E}[\|\widetilde{\Phi}^0(x)\|_2\|\Delta\Phi^0(x)\|_2]\\ &\leq \frac{2C}{\sqrt{n}}\left(\frac{1}{n}\mathbb{E}[\|\widetilde{\Phi}^0(x)\|_2^2]\right)^{1/2} = \frac{2C\widetilde{\Theta}^{1/2}}{\sqrt{n}} = O_{\hat{\mathbb{P}}}(1/\sqrt{n})\,.\end{aligned}$$

Since $\frac{1}{n}\mathbb{E}[\|\Delta\Phi_j^0(x)\|_2^2] \leq C^2/n$, we have that $\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2] - \widetilde{\Theta} \to 0$ uniformly in $\mathcal{B}_J$, in probability with respect to the random initialisation $\hat{\mathbb{P}}$.

We still need to show that $\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1| \to 0$. Again we can decompose the term in Φ^0 and we have

$$\begin{aligned}&\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1|\\ &\qquad = \frac{1}{n}\sum_{j=1}^{n}(\mathbb{E}[\widetilde{\Phi}_j^0(x)^2] + 2\mathbb{E}[\widetilde{\Phi}_j^0(x)\Delta\Phi_j^0(x)] + \mathbb{E}[\Delta\Phi_j^0(x)^2])\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1|\,.\end{aligned}$$

Clearly, for every j we have $\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1| \leq J$, and so we can write

$$\begin{aligned}\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\Phi_j^0(x)^2]\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1| &\leq \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\widetilde{\Phi}_j^0(x)^2]\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1|\\ &\quad + \frac{2J}{n}(\mathbb{E}[|\widetilde{\Phi}^0(x)\cdot\Delta\Phi^0(x)|] + \mathbb{E}[\|\Delta\Phi^0(x)\|_2^2])\end{aligned}$$

uniformly in $\mathcal{B}_J$. We know already that $\frac{1}{n}(2\mathbb{E}[|\widetilde{\Phi}^0(x)\cdot\Delta\Phi^0(x)|] + \mathbb{E}[\|\Delta\Phi^0(x)\|_2^2]) = O_{\hat{\mathbb{P}}}(1/\sqrt{n})$. As for the other term, we have

$$\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\widetilde{\Phi}_j^0(x)^2]\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1| \leq \frac{1}{\sqrt{n}}\left(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\widetilde{\Phi}_j^0(x)^4]\right)^{1/2}\left(\sum_{j=1}^{n}\min_{i=1\dots q}(\Delta\mathfrak{s}_{ij}^1)^2\right)^{1/2}.$$

Using an argument analogous to that in the proof of Proposition 3, we have that $\frac{1}{n}\sum_{j=1}^n \mathbb{E}[\widetilde{\Phi}_j^0(x)^4]$ has a finite limit (in probability wrt $\hat{\mathbb{P}}$). On the other hand, we have

$$\sum_{j=1}^n \min_{i=1\dots q}(\Delta\mathfrak{s}_{ij}^1)^2 \leq \sum_{j=1}^n\sum_{i=1}^q(\Delta\mathfrak{s}_{ij}^1)^2 = \|\Delta\mathfrak{s}^1\|_{F,2}^2 \leq J^2\,.$$

We have thus obtained that $\frac{1}{n}\sum_{j=1}^n \mathbb{E}[\widetilde{\Phi}_j^0(x)^2]\min_{i=1\dots q}|\Delta\mathfrak{s}_{ij}^1| = O_{\hat{\mathbb{P}}}(1/\sqrt{n})$, and so we can conclude that $\Theta = \Omega_{\hat{\mathbb{P}}}(1)$, uniformly in $\mathcal{B}_J$ and in probability wrt the random initialisation $\hat{\mathbb{P}}$.

Now, we will show that $G = O_{\hat{\mathbb{P}}}(1)$. We have

$$G = \frac{1}{n}\sum_{j=1}^n \mathbb{E}[\|X_j\|_2^3] \leq \frac{4}{n}\sum_{j=1}^n \mathbb{E}[\|\widetilde{X}_j\|_2^3] + \frac{4}{n}\sum_{j=1}^n \mathbb{E}[\|\Delta X_j\|_2^3]\,.$$

Let us write $X_j = U_j + V_j$, with $U_j = (\zeta_{ij}^1\Phi_j^0(x))_{i=1\dots q}$ and $V_j = (\mathfrak{m}_{ij}^1(\Phi_j^0(x) - \mathbb{E}[\Phi_j^0(x)]))_{i=1\dots q}$. Then $\|\Delta X_j\|_2^3 \leq 4(\|\Delta U_j\|_2^3 + \|\Delta V_j\|_2^3)$.

First, denoting as ζ_j^1 and $\Delta\mathfrak{s}_j^1$ the vectors $(\zeta_{ij}^1)_{i=1\dots q}$ and $(\Delta\mathfrak{s}_{ij}^1)_{i=1\dots q}$, we can write

$$\Delta U_j = \Delta\Phi_j^0(x)\zeta_j^1 + \widetilde{\Phi}_j^0(x)\Delta\mathfrak{s}_j^1 \odot \zeta_j^1 + \Delta\Phi_j^0(x)\Delta\mathfrak{s}_j^1 \odot \zeta_j^1\,,$$

where $\odot$ represents the Hadamart product. Φ^0 and ζ^1 are independent and $\mathbb{E}[|\zeta|^3] = 2\sqrt{2/\pi} < 2$ for $\zeta \sim \mathcal{N}(0,1)$, so we have

$$\mathbb{E}[\|\Delta U_j\|_2^3] \leq 54(q^{3/2}\mathbb{E}[|\Delta\Phi_j^0(x)|^3] + \mathbb{E}[|\widetilde{\Phi}_j^0(x)|^3]\mathbb{E}[\|\Delta\mathfrak{s}_j^1\|_2^3] + \mathbb{E}[|\Delta\Phi_j^0(x)|^3]\mathbb{E}[\|\Delta\mathfrak{s}_j^1\|_2^3])\,.$$

Using that $\|\Delta\Phi^0(x)\|_3^3 \leq \|\Delta\Phi^0(x)\|_2^3 \leq C^3$, we have that

$$\frac{1}{n}\sum_{j=1}^n q^{3/2}\mathbb{E}[|\Delta\Phi_j^0(x)|^3] \leq \frac{q^{3/2}C^3}{n}$$

uniformly in $\mathcal{B}_J$. Then we can notice that $\|\Delta\mathfrak{s}_j^1\|_2 \leq \|\mathbf{1}_q\|_3\|\Delta\mathfrak{s}_j^1\|_6 = q^{1/3}\|\Delta\mathfrak{s}_j^1\|_6$ by Hölder's inequality. Hence

$$\begin{aligned}\frac{1}{n}\sum_{j=1}^n \mathbb{E}[|\widetilde{\Phi}_j^0(x)|^3]\mathbb{E}[\|\Delta\mathfrak{s}_j^1\|_2^3] &\leq \frac{q\|\Delta\mathfrak{s}^1\|_{F,6}^3}{\sqrt{n}}\left(\frac{1}{n}\sum_{j=1}^n \mathbb{E}[|\widetilde{\Phi}_j^0(x)|^6]\right)^{1/2}\\ &\leq \frac{qJ^3}{\sqrt{n}}\left(\frac{1}{n}\sum_{j=1}^n \mathbb{E}[|\widetilde{\Phi}_j^0(x)|^6]\right)^{1/2} = O_{\hat{\mathbb{P}}}(1/\sqrt{n})\end{aligned}$$

uniformly in $\mathcal{B}_J$, where the last equality comes from the usual argument that $\frac{1}{n}\sum_{j=1}^n \mathbb{E}[|\widetilde{\Phi}_j^0(x)|^6]$ has a finite limit in probability (with respect to the random initialisation).

Finally, we can notice that $|\Phi_j^0(x)| \leq \|\Phi^0(x)\|_2 \leq C$ for all j, so that

$$\frac{1}{n}\sum_{j=1}^n \mathbb{E}[|\Delta\Phi_j^0(x)|^3]\mathbb{E}[\|\Delta\mathfrak{s}_j^1\|_2^3] \leq \frac{q^{1/2}}{n}C^3\|\Delta\mathfrak{s}^1\|_{F,3}^3 \leq \frac{q^{1/2}C^3J^3}{n}$$

uniformly in $\mathcal{B}_J$, where we used that $\|\Delta\mathfrak{s}^1_j\|_2 \le \|\mathbf{1}_q\|_6\|\Delta\mathfrak{s}^1_j\|_3 = q^{1/6}\|\Delta\mathfrak{s}^1_j\|_3$ by Hölder's inequality, and that $\|\Delta\mathfrak{s}^1\|_{F,3} \le \|\Delta\mathfrak{s}^1\|_{F,2} \le J$. We can hence conclude that

$$\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\|\Delta U_j\|_2^3] = O_{\hat{\mathbb{P}}}(1/\sqrt{n})$$

uniformly in $\mathcal{B}_J$.

Now we need to bound $\|\Delta V_j\|_2$. Letting $\mathfrak{m}^1_j = (\mathfrak{m}^1_{ij})_{i=1\dots q}$ and $\delta\Phi^0_j(x) = \Phi^0_j(x) - \mathbb{E}[\Phi^0_j(x)]$, it can be easily shown that

$$\|\Delta V_j\|_2 \le |\delta\widetilde{\Phi}^0_j(x)|\|\Delta\mathfrak{m}^1_j\|_2 + |\Delta\delta\Phi^0_j(x)|\|\widetilde{\mathfrak{m}}^1_j\|_2 + \|\Delta\mathfrak{m}^1_j\|_2|\Delta\delta\Phi^0_j(x)|\,.$$

So we have

$$\begin{aligned}\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\|V_j\|_2^3] \le{}& \frac{27}{n}\sum_{j=1}^{n}\mathbb{E}[|\delta\widetilde{\Phi}^0_j(x)|^3]\|\Delta\mathfrak{m}^1_j\|_2^3\\ &+\frac{27}{n}\sum_{j=1}^{n}\mathbb{E}[|\Delta\delta\Phi^0_j(x)|^3]\|\widetilde{\mathfrak{m}}^1_j\|_2^3 + \frac{27}{n}\sum_{j=1}^{n}\mathbb{E}[|\Delta\delta\Phi^0_j(x)|^3]\|\Delta\mathfrak{m}^1_j\|_2^3\,.\end{aligned}$$

Starting from the first term, we have that

$$\sum_{j=1}^{n}\mathbb{E}[|\delta\widetilde{\Phi}^0_j(x)|^3]\|\Delta\mathfrak{m}^1_j\|_2^3 \le \left(\sum_{j=1}^{n}\mathbb{E}[|\delta\widetilde{\Phi}^0_j(x)|^6]\right)^{1/2}\left(\sum_{j=1}^{n}\|\Delta\mathfrak{m}^1_j\|_2^6\right)^{1/2}.$$

From Hölder's inequality we have $\|\Delta\mathfrak{m}^1_j\|_2 \le \|\mathbf{1}_q\|_3\|\Delta\mathfrak{m}_j\|_6 = q^{1/3}\|\Delta\mathfrak{m}_j\|_6$ and hence

$$\begin{aligned}\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[|\delta\widetilde{\Phi}^0_j(x)|^3]\|\Delta\mathfrak{m}^1_j\|_2^3 &\le \frac{q\|\Delta\mathfrak{m}^1\|_{F,6}^3}{\sqrt{n}}\left(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[|\delta\widetilde{\Phi}^0_j(x)|^6]\right)^{1/2}\\ &\le \frac{qJ^3}{\sqrt{n}}\left(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[|\delta\widetilde{\Phi}^0_j(x)|^6]\right)^{1/2} = O_{\hat{\mathbb{P}}}(1/\sqrt{n})\end{aligned}$$

uniformly in $\mathcal{B}_J$, as $\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[|\delta\widetilde{\Phi}^0_j(x)|^6]$ tends in probability (wrt the random initialisation) to a finite limit.

Proceeding analogously, and noting that the L-Lipschitzianity of ϕ implies that $\mathbb{E}[|\Delta\delta\Phi^0_j(x)|^3] \le 8L^3\mathbb{E}[|\Delta Y^0_j(x)|^3]$, we get

$$\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[|\Delta\delta\Phi^0_j(x)|^3]\|\widetilde{\mathfrak{m}}^1_j\|_2^3 \le \frac{8C^3}{\sqrt{n}}\left(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\|\widetilde{\mathfrak{m}}^1_j\|_2^6]\right)^{1/2} = O_{\hat{\mathbb{P}}}(1/\sqrt{n})$$

uniformly in $\mathcal{B}_J$, and again we used the fact that $\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\|\widetilde{\mathfrak{m}}^1_j\|_2^6]$ converges in probability (wrt the random initialisation) to a finite quantity to show that the above expression is of order $O_{\hat{\mathbb{P}}}(1/\sqrt{n})$.

Finally, in a similar way we get

$$\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[|\Delta\delta\Phi_j^0(x)|^3]\|\Delta\mathfrak{m}_j^1\|_2^3 \leq \frac{8qJ^3C^3}{n}$$

uniformly in $\mathcal{B}_J$. We can hence conclude that

$$\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[\|V_j\|_2^3] = O_{\hat{\mathbb{P}}}(1/\sqrt{n})$$

and so that, as $n \to \infty$, $G \leq O_{\hat{\mathbb{P}}}(1)$, uniformly in $\mathcal{B}_J$ and in probability with respect to the random initialisation. This ends the proof. ■

Appendix B. Multiclass classification ($q > 2$)

In the framework of Section 4.1, things get more complicated when there are more than two classes. We can write

$$\mathbb{E}[\ell(\hat{f}(x), i^\star)] = \mathbb{P}\left(F_{i^\star}(x) \leq \max_{i\neq i^\star} F_i(x)\right) = 1 - \mathbb{P}\left(F_{i^\star}(x) > \max_{i\neq i^\star} F_i(x)\right)\,.$$

Hence, given a q-dimensional Gaussian vector $Y \sim \mathcal{N}(M, Q)$, we need to find an estimate of $\mathbb{P}(Y_{i^\star} > \max_{i\neq i^\star} Y_i)$.

The most trivial estimator would consist of sampling different realisations of Y and then give a MC estimate. However, as we are interested in the gradient of the expected loss, this method will not work. Indeed, the gradient of this estimate is the sum of the gradients of the 01-loss of each sample. As all these gradients are null, we do not obtain anything informative. We have thus to proceed in a less naive way.

Let us assume that $i^\star = q$ (the largest label). Hence, we will focus on $\mathbb{P}(Y_q > \max_{i<q} Y_i)$. With a Cholesky-like algorithm, we can find a lower triangular matrix A such that $Y \sim AX + M$, where $X \sim \mathcal{N}(0, \mathrm{Id})$. We have $Y_i = \sum_{i'=1}^{q} A_{ii'}X_{i'} + M_i$ and $A_{iq} = 0$ for $i = 1 \dots (q-1)$, while $A_{qq} > 0$. For $i < q$, we can write

$$\mathbb{P}(Y_q > Y_i) = \mathbb{P}\left(X_q > \sum_{i'=1}^{q-1}\frac{A_{ii'} - A_{qi'}}{A_{qq}}\,X_{i'} + \frac{M_i - M_q}{A_{qq}}\right)\,.$$

Let us define the $(q-1)$ dimensional random vector $\tilde{X}$ as $\tilde{X} = \tilde{A}X + \tilde{M}$, where $\tilde{A}$ is a $(q-1)\times q$ matrix and $\tilde{M}$ is a $(q-1)$ vector, whose elements are given by $\tilde{A}_{ii'} = \frac{A_{ii'}-A_{qi'}}{A_{qq}}$ and $\tilde{M}_i = \frac{M_i - M_q}{A_{qq}}$ repectively. With this notation, we have $\mathbb{P}(Y_q > Y_i) = \mathbb{P}(X_q > \tilde{X}_i)$. Now, we have gained that X_q is independent from all the other X_i's, and so from all the $\tilde{X}_i$'s. In short, $(X_q|\tilde{X}) = X_q \sim \mathcal{N}(0,1)$. So, we can write

$$\mathbb{P}\left(Y_q > \max_{i<q} Y_i\right) = \mathbb{P}\left(X_q > \max_{i<q}\tilde{X}_i\right) = \mathbb{E}\left[\mathbb{P}\left(X_q > \max_{i<q}\tilde{X}_i\middle|\tilde{X}\right)\right]\,.$$

Now, if we let $\psi(u) = \frac{1}{2}(1 - \text{erf}\,(u/\sqrt{2}))$, we get

$$\mathbb{P}\left(Y_q > \max_{i<q} Y_i\right) = \mathbb{E}\left[\psi\left(\max_{i<q} \tilde{X}_i\right)\right] .$$

We can estimate the above expression with MC sampling. Note that it is almost everywhere differentiable with respect to the components of M and Q (as the Cholesky transform is differentiable) and the gradient with respect to M and Q is not trivially null.

Finally, for a general $i^\star \in \{1, \ldots, q\}$, we can get $\mathbb{P}(Y_{i^\star} > \max_{i \neq i^\star} Y_i)$ by simply performing a swap of the two labels $i^\star$ and q, and then apply the method for $i^\star = q$.

Appendix C. Expected values for ReLU and sin activations

Let $a > 0$, $b \in \mathbb{R}$, $\zeta \sim \mathcal{N}(0, 1)$. The following formulae are easily verified by direct calculations:

$$\begin{aligned}
&\mathbb{E}[\sin(a\zeta + b)] = e^{-a^2/2} \sin b\,;\\
&\mathbb{E}[\sin(a\zeta + b)^2] = \frac{1}{2}(1 - e^{2a^2} \cos(2b))\,;\\
&\mathbb{E}[\mathrm{ReLU}(a\zeta + b)] = \frac{a e^{-b^2/(2a^2)}}{\sqrt{2\pi}} + \frac{b}{2}\left(1 + \text{erf}\,\frac{b}{a\sqrt{2}}\right)\,;\\
&\mathbb{E}[\mathrm{ReLU}(a\zeta + b)^2] = \frac{ab e^{-b^2/(2a^2)}}{\sqrt{2\pi}} + \frac{1}{2}(a^2 + b^2)\left(1 + \text{erf}\,\frac{b}{a\sqrt{2}}\right)\,.
\end{aligned}$$

Appendix D. Experimental details

In all the experiments, the training consisted of optimising some PAC-Bayesian bound via SGD with momentum parameter 0.9. The PAC parameter δ was always chosen equal to 0.025. We only performed the training of the means $\mathfrak{m}$ and all the networks considered had no bias. The priors corresponded to the initialisation of the network (6). Note that in our implementation, the scaling factors $1/\sqrt{p}$ and $1/\sqrt{n}$ were absorbed in the hyper-parameters, so that we performed the gradient descent on $\mu^0 = \mathfrak{m}^0/\sqrt{p}$ and $\mu^1 = \mathfrak{m}^1/\sqrt{n}$ (the standard deviations were kept fixed).

For the binary MNIST experiments, the digits from 0 to 4 were relabelled as 0 and those from 5 to 9 as 1. The training dataset used was the standard one for MNIST, consisting of $m = 60000$ datapoints. For the "standard" PAC-Bayesian methods, the objectives used are those reported in (11). For the objective `lbd` we proceeded by alternating the optimisation of the network hyper-parameters with that of λ, as in Pérez-Ortiz et al. (2021), always enforcing $\lambda \in (0, 1)$. The "Gaussian" training was performed with the optimisation objective (12). All of these methods were used to train the same stochastic network, initialised as in (6). We tried two different learning rate (LR) schedules, the first consisting of 10000 epochs with LR $\eta = 10^{-5}$ and the second of 100 epochs with $\eta = 10^{-2}$, followed by 1000 epochs with $\eta = 10^{-3}$ and 5000 epochs with $\eta = 10^{-4}$. In Table 1 in the main text we report the results of the training schedule achieving the tightest bound, that is the multi-LR schedule for `invkl` and `quad`, and the single-LR schedule for `McAll` and `lbd`.

For the full MNIST experiments, again we used the standard training dataset with $m = 60000$ datapoints. For the "standard" methods, $L_S(\mathcal{Q})$ in (11) was a bounded version of the cross-entropy: we fixed $p_0 = 10^{-5}$ and constrained the probabilities in the definition of the cross-entropy to be greater or equal than p_0, see (Pérez-Ortiz et al., 2021) for more details. In this way, the loss is bounded by $\log(1/p_0)$, and by rescaling it of the same factor we can get a loss bounded in $[0, 1]$. $L_S(\mathcal{Q})$ is the empirical average of this quantity on the training dataset. As we previously did for the binary MNIST experiment, during the training we estimated $L_S(\mathcal{Q})$ by sampling once per iteration the hyper-parameters of the network. The “Gaussian” method used the objective (12), where $L_S(\mathcal{Q})$ is the expected empirical 01-loss. As we are dealing with multiclass classification we do not have a simple expression for the 01-loss, so we used the method described in Appendix B. Per each iteration, the loss was evaluated by an MC estimate averaging 10^4 independent realisations. For all the methods, the training consisted of 10000 epochs with learning rate $\eta = 10^{-5}$.