PAC-Bayesian Learning of Aggregated Binary Activated Neural Networks with Probabilities over Representations

Louis Fortier-Dubois[†,*], Benjamin Leblanc[†], Gaël Letarte[†], François Laviolette[†], Pascal Germain[†]
[†] Département d'informatique et de génie logiciel, Université Laval

Abstract

Considering a probability distribution over parameters is known as an efficient strategy to learn a neural network with non-differentiable activation functions. We study the expectation of a probabilistic neural network as a predictor by itself, focusing on the aggregation of binary activated neural networks with normal distributions over real-valued weights. Our work leverages a recent analysis derived from the PAC-Bayesian framework that derives tight generalization bounds and learning procedures for the expected output value of such an aggregation, which is given by an analytical expression. While the combinatorial nature of the latter has been circumvented by approximations in previous works, we show that the exact computation remains tractable for deep but narrow neural networks, thanks to a dynamic programming approach. This leads us to a peculiar bound minimization learning algorithm for binary activated neural networks, where the forward pass propagates probabilities over representations instead of activation values. A stochastic counterpart that scales to wide architectures is proposed.

Keywords: Statistical learning, PAC-Bayes, binary activated neural networks, representation learning

1. Introduction

The computation graphs of deep neural networks (*a.k.a.* architectures) are challenging to analyze, due to their multiple composition of non-linearities [1], overparametrization [2] and highly non-convex learning objective [3, 4]. Studying simpler models seems a sensible strategy to gain insight on neural network behaviour and state performance guarantees [*e.g.* 5, 6]. Our work starts from one possible simplification, obtained by considering the binary activation function, meaning that each neuron outputs only one bit of information instead of the many bits needed to represent a real number. The use of neural networks involving binary weights, with or without binary activation [7–9], has been suggested for reducing their resource consumption, and these may be especially useful in view of using a pre-trained network for forward propagation on embedded systems, but this is not our primary objective. We aim to foster an atypical vision of neural networks, where binary activated networks with real-valued parameters are viewed as elementary pieces of an ensemble, that we study as a whole.

Our work is motivated by the analysis of Letarte, Germain, Guedj, and Laviolette [10] and Biggs and Guedj [11], rooted in the PAC-Bayes theory [12]: they propose a learning objective for aggregation of binary activated networks derived from a high-confidence generalization bound, and they empirically show that minimizing this objective provides a predictor with both tight and theoretically sound guarantees. They also derived an analytical expression for the expected output value of binary activated networks sampled from Gaussian distributions. Being differentiable, this expression enables gradient descent optimization, and is to be added to methods to train networks with non-differentiable activation functions [13–15]. However, in order to preserve valid PAC-Bayes guarantees and be able to train large neural networks, Letarte et al. [10] rely on an approximation: neurons on the same layer all treat their inputs as if they were independent draws of the same probability distribution, when in fact all

[*] louis.fortier-dubois.1@ulaval.ca

the inputs must correspond to the same draw.[1] Not only does this make the algorithm output values that are slightly straying from the true aggregation expectation, it also fatally increases the PAC-Bayes bound for deeper architectures. Lately, Biggs and Guedj [11] revisited the PAC-Bayes aggregation of neural networks, notably providing lower-variance approximation schemes; doing so, they exhibited that the *forward propagation* function of Letarte et al. can be rewritten in order to remove the above-mentioned independence assumption. Starting from this result, we conduct the major part of our study stepping away from all approximations.

Hence, our first contribution highlights that the exact expectation of an aggregation of binary activated networks is computable in time exponential in the width of the network, and linear in its depth. The originality of our proposed learning algorithm relies on computing probabilities of occurrences of the hidden layer representations. Doing so, not only our algorithm allows us to obtain non-vacuous PAC-Bayes bounds for very deep binary activated networks, but reveals itself as an interesting prediction mechanism. Noteworthy, we show that once the parameters are optimized, the prediction on a new example is achievable with a time complexity that remains *constant* relatively to the network depth. Our second contribution consists in the analysis of a compact scheme of our resulting predictors, having a constant computing time regarding the network depth, and the dichotomy between the first layer of the network versus the following layers. For both these contributions, we present a stochastic version with sub-exponential time complexity regarding the network width.

2. Background and notation

We focus our study on the task of binary classification, using *binary activated multilayer* (BAM) neural networks, *i.e.*, networks where each neuron either outputs -1 or $+1$, using the sign function: $\operatorname{sgn}(x) = -1$ if $x < 0$, $+1$ otherwise. We consider fully connected BAMs of $L \in \mathbb{N}^+$ layers of size $d_k \in \mathbb{N}^+$, $\forall k \in \{1, 2, \dots, L\}$, and inputs of size $d_0 \in \mathbb{N}^+$. We fix $d_L = 1$, whose output is the classification output of the whole network. We call L the *depth* of the network, d_k the *width* of the k^{th} layer, and $\max_{k\in\{1,2,\dots,L\}} d_k$ the *width* of the network. The sequence $\langle d_k \rangle_{k=0}^{L}$ constitutes a (fully connected) *architecture*. Unlike in *binary* neural networks [*e.g.* 7], the parameters of a BAM are not constrained to be binary, but only activations are binary-valued. We thus have weights $\mathbf{W}_k \in \mathbb{R}^{d_k \times d_{k-1}}$ and biases $\mathbf{b}_k \in \mathbb{R}^{d_k}$, for $k \in \{1, 2, \dots, L\}$. That being said, in the remaining, the equations will be stated without loss of generality in terms of the weights $\mathbf{W}_k$ only. Therefore, a BAM $\mathcal{B}$ is totally defined by the tuple $\mathcal{B} \coloneqq \langle \mathbf{W}_k \rangle_{k=1}^{L}$. See Fig. 1a for an example of architecture. The 0^{th} layer is the input layer, the 1^{st} one is the *leading* hidden layer, any k^{th} layer, with $1 < k < L$, is simply called a hidden layer and the L^{th} one is the output layer. Following the premises of Letarte et al. [10], we consider a distribution over BAMs, which we call an aggregation of BAMs.

Definition 1. *An* aggregation of BAMs *with mean parameters $\mathcal{B}_M$, denoted $\mathcal{A}(\mathcal{B}_M)$, is given by an isotropic Gaussian probability distribution over all parameters, centered in $\mathcal{B}_M$.*

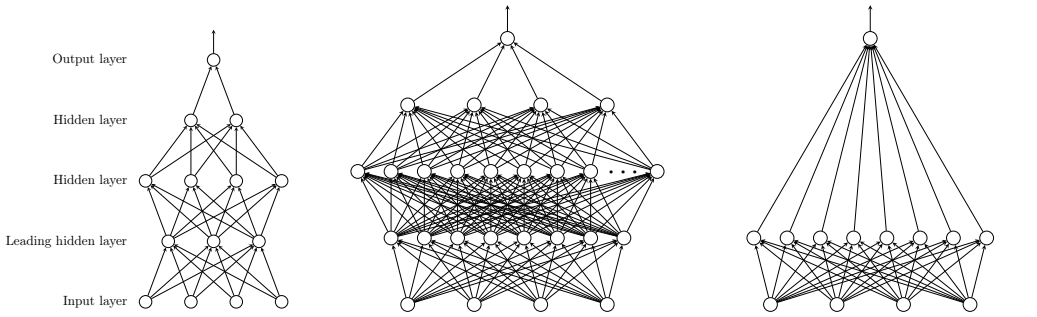
The BAM forward propagation process consists of computing the following function for an input $\mathbf{x} \in \mathbb{R}^{d_0}$, where sgn is defined element-wise:

$$F_{\mathcal{B}}(\mathbf{x}) = \operatorname{sgn}(\mathbf{W}_L \operatorname{sgn}(\mathbf{W}_{L-1} \operatorname{sgn}(\dots \operatorname{sgn}(\mathbf{W}_1 \mathbf{x}) \dots))) \,.$$

In other words, the output of a BAM $\mathcal{B}$ is $F_{\mathcal{B}}(\mathbf{x}) \coloneqq F_{\mathcal{B}}^{L}(\mathbf{x})$, given the recursive equations

$$F_{\mathcal{B}}^{k}(\mathbf{x}) = \begin{cases} \operatorname{sgn}(\mathbf{W}_1 \mathbf{x}) & \text{if } k = 1, \\ \operatorname{sgn}(\mathbf{W}_k F_{\mathcal{B}}^{k-1}(\mathbf{x})) & \text{otherwise.} \end{cases}$$

[1] This recalls the *mean-field approximation* performed by Soudry, Hubara, and Meir [9] for learning binary activated networks in a Bayesian setting.



(a) A typical **BAM** architecture.

(b) The **ABNet** architecture, with underlying BAM from (a).

(c) The **Compact ABNet** structure, obtained from the ABNet of (b).

Figure 1. The BAM shown in (a) has depth $L = 4$, and widths $\langle d_k \rangle_{k=0}^{4} = \langle 4, 3, 4, 2, 1 \rangle$. In (b), hidden layers have a width of 2^{d_k} (*i.e.*, one neuron per binary representation obtainable from a layer width of d_k in the underlying BAM). In (c), all hidden layers are merged, leaving a depth of 2 (as explained in Section 5).

Beside, the output of an *aggregation of BAMs* is the expected output of a BAM drawn from the (continuous) parameters distribution:

$$F_{\mathcal{A}(\mathcal{B}_M)}(\mathbf{x}) = \mathbf{E}_{\mathcal{B} \sim \mathcal{A}(\mathcal{B}_M)} F_{\mathcal{B}}(\mathbf{x}). \tag{2.1}$$

Such aggregation of BAMs is our main object of study. Note that the forthcoming Section 4 is dedicated to a PAC-Bayesian treatment where $\mathcal{A}(\mathcal{B}_M)$ is considered as a *posterior distribution*. In line with the (PAC-)Bayesian literature, we call the single BAM network $\mathcal{B}_M$ the *Maximum-A-Posteriori* (MAP) predictor.

3. The aggregation output

We present a recursive formulation to compute the exact aggregation output, denoted $F_{\mathcal{A}(\mathcal{B}_M)}(\mathbf{x})$, which is differentiable end-to-end. The formulation is equivalent to the one of Biggs and Guedj [11, Eq. (9)], but is expressed in order to highlight the probabilities of representations for each layer, which is the cornerstone of our analysis.

Given a *fixed* representation (an input vector $\mathbf{a} \in \mathbb{R}^d$), the expected output of a single neuron with sign activation over an isotropic Gaussian distribution centered on $\mathbf{w} \in \mathbb{R}^d$ is

$$\mathop{\mathbf{E}}_{\mathbf{v} \sim \mathcal{N}(\mathbf{w}, \mathbf{I})} \operatorname{sgn}(\mathbf{v} \cdot \mathbf{a}) = \operatorname{erf}\left(\frac{\mathbf{w} \cdot \mathbf{a}}{\sqrt{2} \|\mathbf{a}\|} \right),$$

where $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ is the Gauss error function. However, as the input vector of a neuron relies on the distribution of weights on the previous layers, the representation itself is a random vector. A neuron outputs a value $s \in \{-1, 1\}$ with probability

$$\Pr(f_{\mathbf{w}}(\mathbf{a}) = s) = \frac{1}{2} + \frac{s}{2} \operatorname{erf}\left(\frac{\mathbf{w} \cdot \mathbf{a}}{\sqrt{2} \|\mathbf{a}\|} \right). \tag{3.1}$$

Thus, the probability of observing a specific representation $\mathbf{s} = (\mathbf{s}^1, \ldots, \mathbf{s}^{d_k}) \in R_k$ (with $R_k \coloneqq \{-1, 1\}^{d_k}$) at layer k can be expressed in a recursive manner, having the events

$a_k^{\mathbf{s}} \coloneqq F_{\mathcal{B}}^k(\mathbf{x}) = \mathbf{s}$ and $a_{k-1}^{\bar{\mathbf{s}}} \coloneqq F_{\mathcal{B}}^{k-1}(\mathbf{x}) = \bar{\mathbf{s}}$:

$$\Pr(a_k^{\mathbf{s}}) = \begin{cases} \prod_{i=1}^{d_1} \left(\frac{1}{2} + \frac{\mathbf{s}^i}{2} \mathrm{erf}\left(\frac{\mathbf{W}_1^i \cdot \mathbf{x}}{\sqrt{2}\|\mathbf{x}\|} \right) \right) & \text{if } k = 1, \\ \sum_{\bar{\mathbf{s}} \in R_{k-1}} \Pr(a_k^{\mathbf{s}} \mid a_{k-1}^{\bar{\mathbf{s}}}) \Pr(a_{k-1}^{\bar{\mathbf{s}}}) & \text{otherwise.} \end{cases}$$

The base case of the above recursion refers to the leading hidden layer ($F_{\mathcal{B}}^1$). The probability of observing a representation $\mathbf{s} \in R_1$ on this first hidden layer given an input $\mathbf{x}$ amounts to be the product of the probability associated with the d_1 individual neuron values (Eq. 3.1). The general case refers to the subsequent hidden layers ($F_{\mathcal{B}}^2, \ldots, F_{\mathcal{B}}^L$). The probability of observing a representation $\mathbf{s} \in R_k$ is decomposed into the sum of the $2^{d_{k-1}}$ probabilities of observing the representation $\bar{\mathbf{s}} \in R_{k-1}$ on the previous hidden layer (obtained recursively) and the conditional probability (obtained from Eq (3.1), using the fact that $\|\bar{\mathbf{s}}\|^2 = d_{k-1}$) :

$$\Pr(a_k^{\mathbf{s}} \mid a_{k-1}^{\bar{\mathbf{s}}}) = \prod_{i=1}^{d_k} \left[\frac{1}{2} + \frac{\mathbf{s}^i}{2} \mathrm{erf}\left(\frac{\mathbf{W}_k^i \cdot \bar{\mathbf{s}}}{\sqrt{2 d_{k-1}}} \right) \right] .$$

Finally, assuming the output layer is only one neuron wide, the exact output of the aggregation can be computed with: $F_{\mathcal{A}(\mathcal{B}_M)}(\mathbf{x}) = \mathbf{E}_{\mathcal{B} \sim \mathcal{A}(\mathcal{B}_M)} F_{\mathcal{B}}(\mathbf{x}) = \Pr(F_{\mathcal{B}}^L(\mathbf{x}){=}1) - \Pr(F_{\mathcal{B}}^L(\mathbf{x}){=}-1)$.

By contrast, the proposed PBGNet algorithm of Letarte et al. [10, Eq. (16)] computes $F_{PBG}^L(\mathbf{x})$ using the following equations (equivalent to the ones above only when $L \leq 2$):

$$F_{PBG}^k(\mathbf{x}) = \begin{cases} \mathrm{erf}\left(\frac{\mathbf{W}_1 \cdot \mathbf{x}}{\sqrt{2}\|\mathbf{x}\|} \right) & \text{if } k = 1, \\ \sum_{\bar{\mathbf{s}} \in R_{k-1}} \mathrm{erf}\left(\frac{\mathbf{W}_k \cdot \bar{\mathbf{s}}}{\sqrt{2 d_{k-1}}} \right) \prod_{i=1}^{d_k} \left[\frac{1}{2} + \frac{\bar{\mathbf{s}}^i}{2} \left(F_{PBG}^{k-1}(\mathbf{x}) \right)^i \right] & \text{otherwise.} \end{cases}$$

This comes down to outputting at each layer only the expectation of the BAM representation given the expectation of the previous layer, instead of our complete probability distribution. Their method therefore deletes information at each layer since the expectation does not carry the correlation between each individual neuron output. This approximation recalls the mean-field one, on which the Bayesian analysis of binary activated networks of Soudry, Hubara, and Meir [9] relies. We avoid such approximations, and we compute $F_{\mathcal{A}(\mathcal{B}_M)}(\mathbf{x})$ by a dynamic programming approach, described next.

Dynamic program. Abusing notation a little, when writing $\left[g(\mathbf{s}) \right]_{\mathbf{s} \in R}$ we assume all $\mathbf{s}$ are taken in lexicographical order. Hence, posing

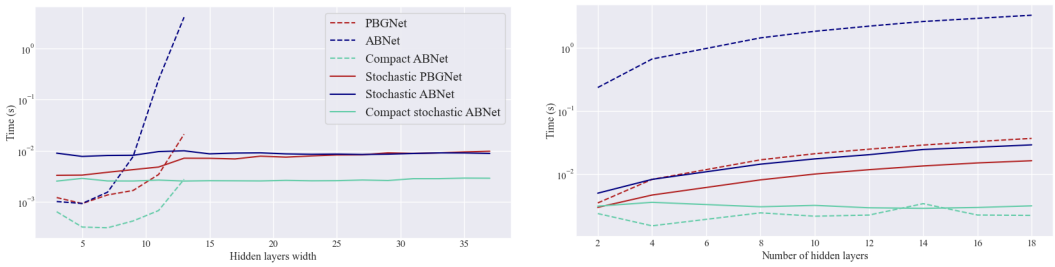
$$\mathbf{\Psi}_k = \left[\prod_{i=1}^{d_k} \left(\frac{1}{2} + \frac{\mathbf{s}^i}{2} \mathrm{erf}\left(\frac{\mathbf{W}_k^i \cdot \bar{\mathbf{s}}}{\sqrt{2 d_{k-1}}} \right) \right) \right]_{\mathbf{s} \in R_k, \bar{\mathbf{s}} \in R_{k-1}} , \tag{3.2}$$

one can obtain straightforwardly the probability vector $\mathbf{P}_k = \left[\Pr(F_{\mathcal{B}}^k(\mathbf{x}) \mathbf{s}) \right]_{\mathbf{s} \in R_k}$ by computing $\mathbf{\Psi}_k \cdot \mathbf{P}_{k-1}$. Starting with

$$\mathbf{P}_1(\mathbf{x}) = \left[\prod_{i=1}^{d_1} \left(\frac{1}{2} + \frac{\mathbf{s}^i}{2} \mathrm{erf}\left(\frac{\mathbf{W}_1^i \cdot \mathbf{x}}{\sqrt{2}\|\mathbf{x}\|} \right) \right) \right]_{\mathbf{s} \in R_1} , \tag{3.3}$$

and computing $\mathbf{P}_k$ for $k \in \{2, 3, \ldots, L\}$ in ascending order, we can therefore compute the exact expectation of a BAM $\mathcal{B} \sim \mathcal{A}$ in time exponential in $\mathcal{B}$'s width, yet linear in its depth.

The previous formulas lead to what stands as the forward propagation process of our new neural network, which we name *ABNet* for ***A**ggregation of **B**inary activated **Net**works*. From parameters $\mathcal{B}_M$, it computes $F_{\mathcal{A}(\mathcal{B}_M)}(\mathbf{x})$. The computation graph of ABNet is illustrated by Fig. 1b. The width of hidden layers $k < L$ are of exponential size 2^{d_k} relatively to the width d_k of the BAM networks it aggregates. Each layer of ABNet outputs a probability



(a) Computation time according to the network width, with depth $L = 6$.

(b) Computation time according to the network depth, with width $d_k = 12$ for $1 \leq k < L$.

Figure 2. Empirical study of the time needed for the forward propagation of our four ABNet versions and the benchmark PBGNet, on a batch of 32 examples of the Ads dataset [16], with 100 samples for stochastic versions, averaged on 100 repetitions. As can be seen in Fig. 2a, the memory requirements of our (non-stochastic) PBGNet and ABNet implementations exceed the available resources for layer widths greater than 13.

distribution over all possible configurations of the underlying BAM. The next layer then multiplies those probabilities by the conditional probabilities $\mathbf{\Psi}$, which is just a reorganization of the weights and is totally independent of the input $\mathbf{x}$. As a result, ABNet applies only linear functions on hidden and output layers; an observation we discuss further in Section 5.

Stochastic version. ABNet has many interesting theoretical properties, but the necessity of computing the probability of every combination of neuron outputs at a given layer makes it too cumbersome for practical applications. We propose a stochastic version of ABNet, which keeps its property of avoiding the mean field approximation while limiting the computation complexity with regard to the width to a quadratic one. Note that the stochastic versions of ABNet and PBGNet are truly different: while the former propagates probabilities over representations, the latter relies on forward and backward passes alike standard neural networks. This is achievable by picking a constant number n of representations R'_k uniformly from R_k at layer k, computing only the occurring probability of those n representations (replacing representation sets R_k by their uniformly drawn counterparts R'_k in Equations 3.2 and 3.3), and normalizing at each layer by dividing $\mathbf{P}_k$ by $\sum_{\mathbf{s}_k \in R'_k} \mathbf{P}_k[\mathbf{s}_k]$.

Complexity. Assuming every layer has the same width d at each layer, the complexity of PBGNet is $\mathcal{O}(L2^d d^2)$ (or $\mathcal{O}(Lnd^2)$ for its stochastic counterpart, with n samples), while the complexity of ABNet is $\mathcal{O}(L2^{2d} d^2)$ (or $\mathcal{O}(Ln^2 d^2)$ for the stochastic version). See Fig. 2 for an empirical study of computing times. A salient fact is that stochastic versions scale much better on wide architectures (Fig. 2a).

4. Bounding and optimizing the generalization loss

Initiated by McAllester [12], the PAC-Bayes theory allows one to bound the generalization error of a learned predictor without requiring a validation set, under the sole assumption that data is sampled in an *iid* way from the unknown distribution $\mathcal{D}$.

To be eligible for a PAC-Bayesian treatment, predictors must be expressed through a *posterior* probability distribution over a predefined class of *hypotheses*. Even though neural networks are not naturally defined as such, many valuable analyses have been proposed by applying a PAC-Bayesian theory to stochastic variants of deterministic neural networks [17–21] by considering perturbations (typically Gaussian distributed noise) on the weights. This strategy can be applied to any neural network topology and activation functions, but the generalization bounds do not apply to the underlying deterministic (non-perturbed)

network. Other theoretical frameworks than PAC-Bayes also leverage probabilistic views of neural networks, typically studying convergence of infinitely wide networks [22, 23].

By adopting the construction of Letarte et al. [10] and designing our predictor *natively* as a distribution over (finite) BAM networks (Eq 2.1), the PAC-Bayesian bound applies to the output of ABNet. The forthcoming Theorem 2 provides high confidence upper bound for the *generalization loss* of a learned ABNet, defined as $\mathcal{L}_{\mathcal{D}}(F_{\mathcal{A}}) = \mathbf{E}_{(\mathbf{x},y)\sim\mathcal{D}}\ \ell(F_{\mathcal{A}}(\mathbf{x}), y)$, where $\ell(y', y) = \frac{1}{2}(1 - yy') \in [0, 1]$ is the linear loss for the binary classification problem. The two main quantities involved in the computation of the bound are the *empirical* loss on the learning sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \sim \mathcal{D}^n$,

$$\hat{\mathcal{L}}_S(F_{\mathcal{A}}) = \frac{1}{n}\sum_{i=1}^{n} \ell(F_{\mathcal{A}}(\mathbf{x}_i), y_i)\,, \tag{4.1}$$

and the *Kullback-Leibler (KL) divergence* between the learned parameters (posterior distribution) $\mathcal{B} = \langle \mathbf{W}_k \rangle_{k=1}^L$ and a reference (prior distribution) $\mathcal{B}^p = \langle \mathbf{W}_k^p \rangle_{k=1}^L$ which is independent of the training data.[2] By using isotropic Gaussians for both the prior and the posterior, the KL divergence is easily obtained with

$$\mathrm{KL}(\mathcal{B}\|\mathcal{B}^p) = \frac{1}{2}\sum_{k=1}^{L}\|\mathbf{W}_k - \mathbf{W}_k^p\|^2. \tag{4.2}$$

The PAC-Bayes theorem below is borrowed from Letarte et al. [10], which itself is a variation from a seminal result from Catoni [24], but has the major advantage to directly deal with the trade-off between the empirical loss and the KL divergence (*i.e.*, the value of C in Equation (4.3) is the one minimizing the bound).

Theorem 2. *Given a data independent prior distribution $\mathcal{B}^p$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$ over a realization of the learning sample $S \sim \mathcal{D}^n$, then, for all posterior $\mathcal{B}$:*

$$\mathcal{L}_{\mathcal{D}}(F_{\mathcal{A}}) \quad \leq \quad \inf_{C>0} \left\{ \tfrac{1}{1-e^{-C}} \left(1 - \exp\left[-C\hat{\mathcal{L}}_S(F_{\mathcal{A}}) - \tfrac{1}{n}\left(\mathrm{KL}(\mathcal{B}\|\mathcal{B}^p) + \ln \tfrac{2\sqrt{n}}{\delta} \right) \right] \right) \right\}. \tag{4.3}$$

A salient feature of the PAC-Bayesian bounds is that they are uniformly valid (with probability at least $1 - \delta$) for the whole family of posterior. This is particularly suited for the design of a bound minimization algorithm, as the right-hand side of Equation (4.3) suggests an objective to minimize, and is providing a generalization guarantee even when the optimization procedure does not converge to a global minimum. Thus, we propose to train the ABNet architecture by minimizing the bound given in Theorem 2 by stochastic gradient descent. That is, the following objective is optimized according to parameters $\mathcal{B}$ and $C > 0$:

$$\tfrac{1}{1-e^{-C}} \left(1 - \exp\left[-C\hat{\mathcal{L}}_S(F_{\mathcal{A}}) - \tfrac{1}{n}\left(\mathrm{KL}(\mathcal{B}\|\mathcal{B}^p) + \ln \tfrac{2\sqrt{n}}{\delta} \right) \right] \right). \tag{4.4}$$

Although this objective may appear similar to the ones of previous PAC-Bayes analyses, the proposed ABNet objective differs in two noticeable ways :
(1) The predictor output, and therefore the empirical loss of Equation (4.1), corresponds to the exact BAM expectation and is computed thanks to the forward propagation routine of ABNet, instead of the more classical computational graph of Letarte et al. [10] or the approximation scheme of Biggs and Guedj [11].
(2) The KL divergence of Equation (4.2), acting as a regularization term, does give the same penalty to weights of every layer in ABNet for networks of depth $L > 2$. In contrast, the corresponding term in PBGNet [10] penalizes weights by a growing factor according to the layer depth.

[2] Following the common practice [10, 11, 18, 21], we chose an SGD random initialization as $\mathcal{B}^p$.

5. Compacting the ABNet

Recall from Section 3 that the aggregation output can be computed by the matrix product $F_{\mathcal{A}(\mathcal{B})}(\mathbf{x}) = \big[1, -1\big] \cdot (\mathbf{\Psi}_L(\mathbf{\Psi}_{L-1}(\dots \mathbf{\Psi}_3(\mathbf{\Psi}_2\mathbf{P}_1(\mathbf{x}))\dots)))$, with $\langle \mathbf{\Psi}_k \rangle_{k=2}^{L}$ and $\mathbf{P}_1(\mathbf{x})$ computed from parameters $\mathcal{B} = \langle \mathbf{W}_k \rangle_{k=1}^{L}$ according to Equations (3.2) and (3.3). From this point of view, ABNet simply computes a linear function of the *leading hidden layer representation* $\mathbf{P}_1(\mathbf{x})$, highlighting a limitation of all binary (and discrete-valued) activated neural networks. Indeed, all matrices $\mathbf{\Psi}_k$ are solely based on the weights and do not rely on the input layer. Since there is no activation function between hidden layers, dot product associativity allows us to state the following.

Proposition 3. *The output of an aggregation of BAMs $\mathcal{A}(\mathcal{B})$, where $\mathcal{B}$ has leading hidden layer width of d_1 and arbitrary width for other hidden layers, can be obtained by forward propagating in a compact (with regard to depth) neural network having a leading hidden layer of width 2^{d_1} with* erf *activation and an output layer of width* 1 *with identity activation:*

$$F_{\mathcal{A}}(\mathbf{x}) = \mathbf{H} \cdot \mathbf{P}_1(\mathbf{x})\,, \tag{5.1}$$

where $\mathbf{P}_1(\mathbf{x})$ is a vector of 2^{d_1} elements defining a probability distribution on the outputs of the leading hidden layer of $\mathcal{B}$ on $\mathbf{x}$, and $\mathbf{H} \in [-1, 1]^{2^{d_1}}$ is a vector giving the expected output of the rest of the network given the output of the first layer, such that

$$\mathbf{H} = \big[1, -1\big] \cdot \mathbf{\Psi}_L \cdot \mathbf{\Psi}_{L-1} \cdot \ldots \cdot \mathbf{\Psi}_3 \cdot \mathbf{\Psi}_2\,. \tag{5.2}$$

Since only $\mathbf{P}_1(\mathbf{x})$ changes in function of $\mathbf{x}$, for fixed weights one can numerically precompute $\mathbf{H} \coloneqq \big[h_{\mathbf{s}}\big]_{\mathbf{s}\in R_1}$ once and for all $\mathbf{x}$. In the underlying BAM this is analogous to precomputing the output for every representation outputted by the leading hidden layer. Every entry of $\mathbf{H}$ is a real number between -1 and 1 since it represents an expectation on a BAM output. This observation leads us to the following corollary.

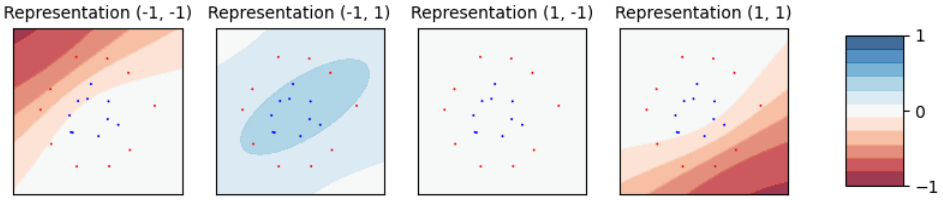
Corollary 4. *Notwithstanding the fact that the underlying BAM architecture can be arbitrarily deep, the aggregation output can always be expressed in the following shallow form, with $h_{\mathbf{s}} \in [-1, 1]$:*

$$F_{\mathcal{A}}(\mathbf{x}) = \sum_{\mathbf{s}\in R_1} h_{\mathbf{s}} \prod_{i=1}^{d_1} \left(\frac{1}{2} + \frac{\mathbf{s}^i}{2} \mathrm{erf}\left(\frac{\mathbf{W}_1^i \cdot \mathbf{x}}{\sqrt{2}\|\mathbf{x}\|} \right) \right). \tag{5.3}$$

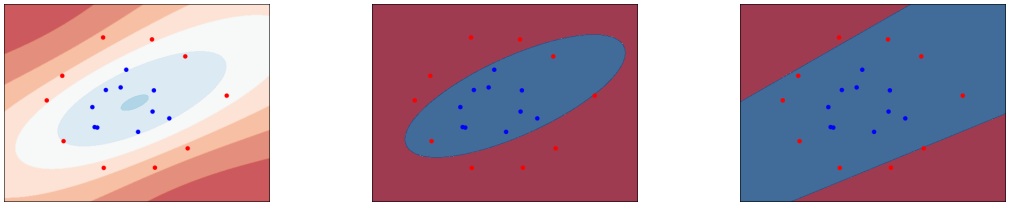
Thus, forward propagation of ABNet can be computed in time constant with regard to L.

We call the algorithm that computes Equation (5.3) the *Compact ABNet*. See Fig. 1c for a visual representation. Interestingly, the PAC-Bayes generalization bound of Theorem 2 is not obtainable directly from the Compact ABNet parameters. Therefore, our bound minimization algorithm requires the ABNet architecture. Noteworthy, our empirical experiments (Section 6, Fig. 4) show that training deeper ABNet can achieve better generalization than a shallower architecture, even when both share a Compact architecture of the same size. Compacting our stochastic version is also possible. Since the dot product must be executed on fixed R'_ks, the drawn samples must be predetermined and remain the same at each inference; this leads to a very concise classifier which performs just as well as the last learned Stochastic ABNet. As can be seen on Fig. 2b, compact networks are much faster at inference time than their deep equivalent, as their complexity does not increase with depth.

Our original approach of propagating probabilities over representations is what brings the light on the compactability phenomenon. It is a well-known result that any function can be approximated to an arbitrary level of accuracy with a neural network having as few as *one* hidden layer, given that the layer is wide enough [25]. It has also been shown that a shallow "student" network can learn to mimic a deep "teacher" network to reach the same performance level [1]. However, typical neural networks do not allow such an explicit



(a) For all $\mathbf{x}$, the probabilities of outputting a specific representation at the leading hidden layer, obtained from $\mathbf{P}_1(\mathbf{x})$, multiplied by the expected output for each of these regions, given by $\mathbf{H}$.



(b) The aggregation output $F_{\mathcal{A}(\mathcal{B}_M)}(\mathbf{x})$ for all $\mathbf{x}$, obtained by summing over representations.

(c) Using ABNet as a classifier, one can simply use $\operatorname{sgn}(F_{\mathcal{A}(\mathcal{B}_M)}(\mathbf{x}))$ as the prediction for an input $\mathbf{x}$.

(d) Using the BAM classifier from the MAP $F_{\mathcal{B}_M}(\mathbf{x})$, the decision boundary is less expressive.

Figure 3. Predictions of an ABNet and its underlying BAM with architecture $\langle 2, 2, 2, 1\rangle$, *i.e.*, with two-dimensional inputs and two hidden layers of two neurons, on a toy dataset.

construction that maps an initially (non-linear) deep structure to a shallow form. The result of Proposition 3 is a curiosity that is worth analyzing further. Remarkably, there is a clear dichotomy between the roles of $\mathbf{P}_1(\mathbf{x})$ and $\mathbf{H}$: the former transforms data points into a probability distribution over the leading hidden layer representations, whereas the latter gives the aggregation output for each of those representations. Put otherwise, the first layer serves as an embedding and the rest of the layers operate as a classifier. Fig. 3 illustrates the particularity of the prediction mechanism of ABNet.

The leading hidden layer. Equation (5.1) implies that the leading hidden layer defines regions in the input space. All subsequent hidden layers together express the output value of these regions. Fig. 3a shows how those regions divide the input space on the toy problem. Each region is associated with one of the four leading hidden layer binary representation.

Each neuron of the leading hidden layer of a BAM defines a hyperplane in the input space, where inputs on one side are mapped to -1, and 1 on the other side. Considering all regions that are enclosed between the hyperplanes yields up to 2^{d_1} regions, corresponding to the 2^{d_1} output representations R_1. Many of those regions may stray very far from the actual data. For example, in Fig. 3a the region corresponding to representation $(1, -1)$ exists on the other side of where the two planes meet, which is far from any existing data[3].

Additional hidden layers. The vector $\mathbf{H}$ represents by extension a function from $\{-1, 1\}^{d_1}$ to $[-1, 1]$. Its role is to determine what sign should be outputted for each region defined by the leading hidden layer, with a confidence term. Its content is not arbitrary since it must be obtained from the weights of the subsequent hidden layers as in Equation (5.2). Depth therefore adds expressivity to BAM aggregations by allowing regions created by the leading hidden layer to output uncorrelated signs, with more or less confidence.

As illustrated by Figures 3c-d, taking the output of ABNet is not equivalent as taking the output of its associated MAP. For the same parameters, the aggregation allows more complex

[3]By considering the few most important region, one could potentially *interpret* ABNet predictions more easily than for classical neural networks. We consider this question as future work, and refer the reader to Montúfar, Pascanu, Cho, and Bengio [26] for a study of regions in the broader context of neural networks with continuous activations.

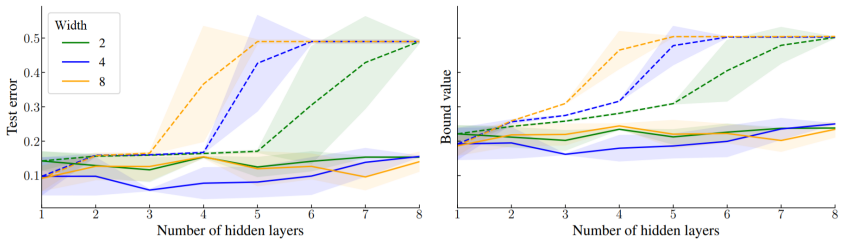


Figure 4. Impact of depth for **PBGNet (dashed)** and **ABNet (solid)** on test errors and bound values according to the width for mnistLH datasets. Results correspond to means and standard deviations over 5 repetitions.

regions than BAMs, taking advantage that an input can belong to several regions, with certain probabilities; there exist points $\mathbf{x}$ and parameters $\mathcal{B}_M$ for which $\text{sgn}(F_{\mathcal{A}(\mathcal{B}_M)}(\mathbf{x})) \neq F_{\mathcal{B}_M}(\mathbf{x})$. For instance, many incorrectly classified -1 data points in Fig. 3d fall within the correct region in Fig. 3c because ABNet can compensate the proximity to the central $+1$ region with a lesser proximity with *two* -1 regions. It is therefore worthy to use the aggregation as a predictor by itself instead of its MAP, for its expressive power. Indeed, Zhu, Dong, and Su [27] observe that ensemble methods on binary neural networks confer stability with regard to input and parameter perturbations, which leads to better generalization.

6. Numerical experiments

We evaluated our proposed approach ABNet by following the experimental framework of Letarte, Germain, Guedj, and Laviolette [10], on the same six binary classification datasets: *ads* and *adult* from the UCI repository [16], along with four MNIST [28] binary variants *mnistLH* (labels $\{0,1,2,3,4\}$ form the "Low" class, and $\{5,6,7,8,9\}$ the "High" class), *mnist17*, *mnist49* and *mnist56* (only examples labeled respectively 1&7, 4&9, and 5&6 are retained).[4] As the exact versions of PBGNet and ABNet are limited by their exponential complexity regarding their width, we explored narrow network architectures (widths $d \in \{2,4,8\}$) and wider architectures (widths $d \in \{10,50,100\}$) accessible only to stochastic versions, all for 1 to 3 hidden layers. All experiments were repeated with 5 different random train/test dataset splits and weights initializations. Networks parameters are optimized using Adam [29] for the following learning rate values: $\{0.1, 0.01, 0.001, 0.0001\}$. Training is performed for 100 epochs with early stopping after 20 epochs without improvement.

We first compare ABNet to its direct counterpart PBGNet [10], both directly optimizing the PAC-Bayesian generalization bound during learning, with the prior distribution defined by the network weights random initialization. We also explore the minimization of the empirical loss with the variants ABNet$_\ell$ (Eq. 4.1) and PBGNet$_\ell$, where 20% of the training data is used as a validation set for model selection.

Even if our primary focus is on the learning of a BAM aggregation, the optimization procedure of PBGNet and ABNet may be used to learn a single BAM, as it is not itself learnable with standard gradient descent methods. We thus compare the *Maximum-A-Posteriori* (MAP) networks of both aggregated methods to three algorithms of the literature for learning neural networks with binary weights and/or activations: *Expectation Backpropagation* [9] (EBP) with real-valued weights and binary activations, *Binarized Neural Network* [8] (BNN) with both binary weights and activations, and *BinaryConnect* [7] (BC) with binary weights

[4]We conducted PBGNet experiments using the publicly available source code of Letarte et al. [10]. We did not perform experiments using the approximation scheme of Biggs and Guedj [11] as their code has not been made public. Moreover, the experiments of the latter were performed solely on *mnistLH* dataset, showing similar accuracy than PBGNet for a similar architecture.

Table 1. Experiment results. On narrow architectures (left column), standard versions of PBGNet and ABNet are used, while their stochastic versions are used on wide architectures (right column). For each dataset and model, the best-performing set of parameters over the repetitions is retained. Shown are the number of hidden layers ($L-1$), hidden size (d), bound value and error rate on the train data (Error_S) and on the test data for the model (Error_T) and the associated MAP BAM. The standard deviation over the five repetitions is shown for the test error. For both BC and BNN algorithms, the Error_T and MAP columns share the same results, as these models don't rely on aggregation.

		Narrow architectures						Wide architectures					
Dataset	Model	$L-1$	d	Bound	Error_S	Error_T	MAP	$L-1$	d	Bound	Error_S	Error_T	MAP
ads	PBGNet	3	2	0.192	0.140	0.141 ± 0.012	0.141	3	10	0.213	0.140	0.141 ± 0.010	0.141
	ABNet	3	2	0.192	0.140	0.141 ± 0.012	0.141	2	10	0.216	0.140	0.141 ± 0.010	0.141
	PBGNet_ℓ	3	4	1.000	0.018	0.026 ± 0.004	0.027	3	10	1.000	0.020	0.026 ± 0.006	0.028
	ABNet_ℓ	3	4	0.887	0.015	0.026 ± 0.003	0.026	2	50	1.000	0.020	0.026 ± 0.005	0.025
	EBP	2	2	–	0.003	0.040 ± 0.008	0.054	3	10	–	0.005	0.035 ± 0.006	0.049
	BC	1	4	–	0.025	0.031 ± 0.004	0.031	1	10	–	0.021	0.032 ± 0.005	0.032
	BNN	1	8	–	0.037	0.038 ± 0.004	0.038	1	100	–	0.029	0.032 ± 0.005	0.032
adult	PBGNet	1	2	0.208	0.157	0.160 ± 0.003	0.158	1	10	0.216	0.156	0.159 ± 0.002	0.158
	ABNet	1	2	0.208	0.157	0.160 ± 0.003	0.158	1	10	0.216	0.156	0.160 ± 0.002	0.158
	PBGNet_ℓ	3	4	0.723	0.135	0.149 ± 0.002	0.156	2	10	0.360	0.146	0.151 ± 0.002	0.164
	ABNet_ℓ	3	4	0.780	0.132	0.149 ± 0.003	0.150	3	10	0.541	0.143	0.151 ± 0.002	0.151
	EBP	1	8	–	0.145	0.152 ± 0.003	0.166	2	100	–	0.049	0.186 ± 0.001	0.189
	BC	1	8	–	0.142	0.151 ± 0.002	0.151	1	50	–	0.160	0.164 ± 0.001	0.164
	BNN	1	2	–	0.180	0.182 ± 0.017	0.182	1	100	–	0.157	0.165 ± 0.002	0.165
mnist17	PBGNet	1	2	0.036	0.005	0.006 ± 0.001	0.006	1	10	0.041	0.005	0.006 ± 0.001	0.006
	ABNet	1	2	0.036	0.005	0.006 ± 0.001	0.006	1	10	0.041	0.005	0.006 ± 0.001	0.006
	PBGNet_ℓ	3	4	1.000	0.002	0.005 ± 0.001	0.005	2	10	1.000	0.002	0.005 ± 0.001	0.006
	ABNet_ℓ	3	2	0.829	0.001	0.004 ± 0.001	0.004	3	10	0.607	0.002	0.005 ± 0.001	0.005
	EBP	1	2	–	0.000	0.006 ± 0.001	0.006	2	10	–	0.000	0.005 ± 0.000	0.005
	BC	2	4	–	0.004	0.010 ± 0.002	0.010	3	50	–	0.003	0.006 ± 0.001	0.006
	BNN	1	8	–	0.003	0.008 ± 0.001	0.008	1	100	–	0.004	0.007 ± 0.001	0.007
mnist49	PBGNet	1	2	0.136	0.036	0.036 ± 0.004	0.036	1	10	0.149	0.037	0.037 ± 0.004	0.036
	ABNet	1	2	0.136	0.036	0.036 ± 0.004	0.036	1	10	0.147	0.038	0.037 ± 0.004	0.036
	PBGNet_ℓ	2	4	1.000	0.008	0.020 ± 0.004	0.020	1	50	0.992	0.004	0.012 ± 0.003	0.012
	ABNet_ℓ	3	8	1.000	0.004	0.017 ± 0.003	0.017	3	10	1.000	0.024	0.029 ± 0.003	0.027
	EBP	3	8	–	0.020	0.033 ± 0.003	0.034	2	10	–	0.001	0.021 ± 0.004	0.026
	BC	2	8	–	0.007	0.016 ± 0.002	0.016	1	100	–	0.005	0.015 ± 0.003	0.015
	BNN	1	2	–	0.030	0.037 ± 0.003	0.037	1	100	–	0.011	0.023 ± 0.003	0.023
mnist56	PBGNet	1	2	0.084	0.021	0.023 ± 0.006	0.023	1	10	0.090	0.023	0.025 ± 0.003	0.024
	ABNet	1	2	0.084	0.021	0.023 ± 0.006	0.023	1	10	0.090	0.023	0.025 ± 0.003	0.024
	PBGNet_ℓ	2	8	1.000	0.004	0.011 ± 0.003	0.011	1	50	0.974	0.003	0.008 ± 0.002	0.008
	ABNet_ℓ	3	8	0.999	0.004	0.009 ± 0.002	0.009	1	10	1.000	0.010	0.017 ± 0.002	0.016
	EBP	3	8	–	0.001	0.010 ± 0.004	0.015	2	10	–	0.000	0.019 ± 0.004	0.021
	BC	1	8	–	0.002	0.009 ± 0.004	0.009	3	50	–	0.004	0.010 ± 0.003	0.010
	BNN	1	8	–	0.013	0.023 ± 0.003	0.023	1	100	–	0.004	0.012 ± 0.001	0.012
mnistLH	PBGNet	1	8	0.186	0.091	0.092 ± 0.036	0.093	1	10	0.167	0.058	0.059 ± 0.010	0.060
	ABNet	3	4	0.162	0.056	0.058 ± 0.002	0.059	2	10	0.187	0.087	0.088 ± 0.006	0.087
	PBGNet_ℓ	3	8	1.000	0.018	0.038 ± 0.002	0.047	1	100	0.998	0.006	0.022 ± 0.001	0.024
	ABNet_ℓ	2	8	0.998	0.025	0.042 ± 0.006	0.043	3	10	0.895	0.050	0.060 ± 0.005	0.058
	EBP	3	8	–	0.016	0.043 ± 0.002	0.082	1	100	–	0.001	0.027 ± 0.001	0.032
	BC	2	8	–	0.023	0.035 ± 0.001	0.035	1	100	–	0.013	0.027 ± 0.001	0.027
	BNN	1	2	–	0.123	0.133 ± 0.004	0.133	1	100	–	0.023	0.036 ± 0.001	0.036

but ReLU activations. Experiments involving EBP, BNN or BC are performed using fully connected networks, following the procedure used for ABNet_ℓ and PBGNet_ℓ.

Narrow architectures. The PAC-Bayesian inspired models with empirical loss minimization (PBGNet_ℓ and ABNet_ℓ) obtain competitive error rates (similar to the results achieved by BC using ReLU activations and binary weights). However, the empirical loss minimization procedure lead to non-informative generalization bounds values. When considering the bound for optimization and model selection for PBGNet and ABNet, selected network architectures are smaller with usually a single hidden layer, as the objective function contains a regularization term on the weight values (see Eq. 4.4), and the error rates grow while bound values improve to a relevant level. Also, bound minimization algorithms are far

less prone to *overfitting* than traditional optimization schemes, as their training errors are remarkably close to their testing errors (recall that PBGNet and ABNet are equivalent for one hidden layer). On the larger and harder dataset mnistLH, the narrow ABNet achieves better error rate and bound value than PBGNet by selecting a deeper architecture thanks to its less penalizing KL divergence regularization. On the performances of the MAP induced BAM networks, error rates are usually similar or slightly higher than their aggregated counterpart, implying these approaches are suitable algorithms to learn BAM networks.

Wide architectures. For all algorithms and most datasets, obtained results for wide and narrow binary neural networks are surprisingly similar. This reveals that constraining ABNet's width to compute the exact aggregation output is not a major caveat. In particular, when one seeks tight PAC-Bayesian guarantees, lower complexity of narrow models should be favored. That being said, the proposed stochastic training for ABNet enables scaling to wider networks. While achieving most of the time comparable results to the stochastic PBGNet, the obtained risk on the large mnistLH dataset suggests that the approximation scheme of ABNet may not be as effective as the exact computation.

Deep architectures. A key improvement of ABNet over PBGNet is the KL divergence computation which is not hindered by a growing factor penalizing the weights according to the network depth. This property should allow ABNet to learn deeper networks with tighter generalization bounds, which we investigated on the mnistLH dataset by extending the main experiment up to 8 hidden layers. Results are presented in Figure 4 where the difference of behaviour between the models is clearly highlighted. For a small number of hidden layers, the performances are similar, but as the number of hidden layer grows, bound values for PBGNet sharply rise and test error rates degrade significantly. On the other hand, bound values are relatively stable for ABNet, indicating its potential to learn deep neural network architectures (the minimum bound is achieved for 3 hidden layers of width 4, which adequately indicates the best test error).

7. Conclusion

Many desirable properties stem from considering a PAC-Bayesian analysis of aggregations over binary activated networks. Like the previous approaches on which we build [10, 11], our proposed learning algorithm gives a sensible way to optimize the parameters of such networks, and provides tight bounds on the generalization error of deep architectures. The originality of our work lies in the focus on the exact computation of aggregation of narrow networks, for which we derive an atypical training scheme based on the propagation of probabilities over hidden layer representations. We further extend the analysis to expose the dichotomy between the first layers versus the others. We believe the latter observation is a sensible tool to understand the expressivity conferred by a network's architecture. Pursuing this research direction could contribute to a line of work [30–32] studying the role of depth in neural networks, but in the context of model aggregation. An interesting perspective is to consider more complex distributions over the network parameters.

References

[1] J. Ba and R. Caruana. "Do Deep Nets Really Need to be Deep?" In: *NIPS*. 2014, pp. 2654–2662.

[2] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. "Understanding deep learning requires rethinking generalization". In: *ICLR*. 2017.

[3] Y. N. Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, and Y. Bengio. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *NIPS*. 2014, pp. 2933–2941.

[4] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. "The Loss Surfaces of Multilayer Networks". In: *AISTATS*. Vol. 38. 2015.

[5] S. Arora, N. Cohen, N. Golowich, and W. Hu. "A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks". In: *ICLR*. 2019.

[6] M. Belkin, D. Hsu, and J. Xu. "Two Models of Double Descent for Weak Features". In: *SIAM J. Math. Data Sci.* 2.4 (2020), pp. 1167–1180.

[7] M. Courbariaux, Y. Bengio, and J. David. "BinaryConnect: Training Deep Neural Networks with binary weights during propagations". In: *NIPS*. 2015, pp. 3123–3131.

[8] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. "Binarized Neural Networks". In: *NIPS*. 2016.

[9] D. Soudry, I. Hubara, and R. Meir. "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights." In: *NIPS*. 2014.

[10] G. Letarte, P. Germain, B. Guedj, and F. Laviolette. "Dichotomize and Generalize: PAC-Bayesian Binary Activated Deep Neural Networks". In: *NeurIPS*. 2019, pp. 6869–6879.

[11] F. Biggs and B. Guedj. "Differentiable PAC-Bayes objectives with partially aggregated neural networks". In: *Entropy* 23.10 (2021), p. 1280.

[12] D. McAllester. "Some PAC-Bayesian Theorems". In: *Machine Learning* 37.3 (1999), pp. 355–363.

[13] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3 (May 1992), pp. 229–256.

[14] Y. Bengio, N. Léonard, and A. C. Courville. "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". In: *CoRR* abs/1308.3432 (2013).

[15] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. "Quantized neural networks: Training neural networks with low precision weights and activations". In: *JMLR* 18.1 (2017), pp. 6869–6898.

[16] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[17] J. Langford and R. Caruana. "(Not) bounding the true error". In: *NIPS*. 2001, pp. 809–816.

[18] G. K. Dziugaite and D. M. Roy. "Data-dependent PAC-Bayes priors via differential privacy". In: *NeurIPS*. 2018, pp. 8440–8450.

[19] 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". In: *ICLR*. 2019.

[20] M. Pérez-Ortiz, O. Rivasplata, J. Shawe-Taylor, and C. Szepesvári. "Tighter risk certificates for neural networks". In: *CoRR* abs/2007.12911 (2020).

[21] K. Pitas. "Dissecting Non-Vacuous Generalization Bounds based on the Mean-Field Approximation". In: *ICML*. 2020.

[22] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. "Deep Neural Networks as Gaussian Processes". In: *CoRR* abs/1711.00165 (2017).

[23] A. Jacot, C. Hongler, and F. Gabriel. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *NeurIPS*. 2018, pp. 8580–8589.

[24] O. Catoni. *PAC-Bayesian supervised classification: the thermodynamics of statistical learning*. Vol. 56. Inst. of Mathematical Statistic, 2007.

[25] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[26] G. F. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. "On the Number of Linear Regions of Deep Neural Networks". In: *NIPS*. 2014, pp. 2924–2932.

[27] S. Zhu, X. Dong, and H. Su. "Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit?" In: *CVPR*. 2019, pp. 4923–4932.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[29] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *ICLR*. 2015.

[30] N. Le Roux and Y. Bengio. "Deep belief networks are compact universal approximators". In: *Neural computation* 22.8 (2010), pp. 2192–2207.

[31] L. Szymanski and B. McCane. "Deep, super-narrow neural network is a universal classifier". In: *IJCNN*. IEEE, 2012, pp. 1–8.

[32] P. Kidger and T. J. Lyons. "Universal Approximation with Deep Narrow Networks". In: *COLT*. Vol. 125. 2020, pp. 2306–2327.