Tight PAC-Bayesian generalization error bounds for deep learning

Anonymous Authors[1]

Abstract

Neural networks are highly expressive, yet generalise well. Here we build on a newly proposed PAC-Bayes analysis that depends on the prior over functions, implicitly encoded in the parameter-function map (1). We obtain non-vacuous bounds that correlate well with the true generalization error for a wide range of modern architectures and binary classification tasks. Our bounds are significantly tighter than other published approaches.

1. Introduction

Generalization in deep learning has been the topic of much recent theoretical and empirical research. Among the many findings, we'd like to highlight three:

1. Modern neural networks (with many more parameters than training points) are highly expressive. For example, they are able to fit randomly labelled data (2). Such high expressivity implies that classical analyses of generalization, which look at the worst-performing ERM algorithm for a given hypothesis class (e.g. based on VC/Rademacher) are bound to fail.

2. For wide enough neural networks, generalization depends only weakly on width. In fact, the Gaussian process approximation of neural networks, which becomes exact in the infinite width limit, appears to predict well the asymptotic performance of SGD-trained neural networks as their width growths (3).

3. While there is some dependence on the optimisation algorithm (e.g. SGD versus GD (4; 5)), Good generalisation can be obtained for multiple different optimisation methods, with typically small differences in the generalization error, relative to the bulk of the generalization.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the ICML 2019 Workshop "Understanding and Improving Generalization in Deep Learning". Do not distribute.

These findings imply that generalization error bounds for deep learning should: (i) Be data and algorithm dependent and: (ii) Should not depend strongly on the number of parameters or the choice of optimization algorithm (within reason).

Inspired by the first requirement, many recent works have proposed hypothesis-dependent bounds, based either on margin bounds (6; 7; 8), or PAC-Bayesian bounds (7; 9; 10; 11). Encouragingly, several of these bounds correlated with the true error in some tasks. Nevertheless, most give vacuous bounds ($\gg 1$) on the generalization error which means these bounds alone don't rigorously explain the generalization performance (even if they capture some important behaviour). Another problem is that several of these bounds scale unrealistically with either width or depth.

Only recently have nonvacuous bounds been obtained (9; 12; 10), although their range of applicability is still limited (applying only to stochastic/compressed networks, or not scaling well for larger architectures), and some remain relatively loose.

In this paper, we build on recent work by Valle-Pérez et al (1), where the authors propose a PAC-Bayesian bound based on the prior over functions implicitly encoded on the parameter-function map of a neural network. This bound is data and algorithm dependent, and satisfies the other desiderata considered above. It works in the infinite width limit (which appears to approximate well the behaviour of networks in practice (13)), and approximates the behaviour of the optimization algorithm with a Bayesian ideal. The authors conjectured that many commonly used optimization algorithms (including SGD) may approximate this ideal (1). In the next section, we provide further reasons why this is a reasonable conjecture. We then empirically test the bound, focusing on the following properties:

- *Applicability over models* We show how the bound can be applied to (almost) any neural network architecture, using a Monte Carlo approximation of the kernel, and make new arguments as to why the bound is justified for SGD-trained neural networks (see Section 2). This is in contrast with most previous work, where the bound only applied to compressed or stochastic networks (9; 10), or the bound was too complex to compute for larger architectures (12).

- *Applicability over datasets* The bound is significantly more efficient to compute that those in (12), but less efficient than those in (10). The main bottleneck is the $O(m^3)$ complexity with the size m of the training set, due to the (approximate) marginal likelihood computation. Nevertheless, we are able to apply the bound to a variety of binary classification tasks, with training sets of up to 40k.

- *Tightness and predictive value*. We empirically show that the bound is relatively tight in most architectures / datasets we tried, and correlates well with the true behaviour of the generalization error.

2. Deterministic PAC-Bayesian bound based on the prior over functions

We work in the same set up as (1) and McAllester (14). We consider binary classification, and a space of functions with codomain $\{0, 1\}$, which we call *concepts*. We consider a prior P over the concept space $\mathcal{H}$, and an algorithm which samples concepts according to the Bayesian posterior, with $0-1$ likelihood. We define *generalization error* as the probability of missclassification upon a new sample $\epsilon(c) = \mathbf{P}_{x\sim\mathcal{D}}[c(x) \neq t(x)]$, where t is the target concept. In Appendix A, we prove the following theorem

Theorem 1. *(Deterministic realizable PAC-Bayes theorem) For any distribution P on any concept space $\mathcal{H}$ and any realizable distribution $\mathcal{D}$ on a space of instances we have, for $0 < \delta \leq 1$, and $0 < \gamma \leq 1$, that with probability at least $1-\delta$ over the choice of sample S of m instances, that with probability at least $1-\gamma$ over the choice of c:*

$$\ln\left(1-\epsilon(c)\right) < \frac{\ln\frac{1}{P(C(S))} + \ln m + \ln\frac{1}{\delta} + \ln\frac{1}{\gamma}}{m-1}$$

where c is chosen according to the posterior distribution $Q(c) = \frac{P(c)}{\sum_{c\in C(S)} P(c)}$, $C(S)$ is the set of concepts in $\mathcal{H}$ consistent with the sample S, and where $P(C(S)) = \sum_{c\in C(S)} P(c)$

The proof is presented in Appendix A. It is follows that of the original PAC-Bayesian theorem by McAllister very closely, with the main technical step relying on the quantifier reversal lemma of McAllister (14). Note that the bound is essentially the same as that of (15), except for the fact that it holds in probability and it adds an extra term dependent on the confidence parameter γ, which is usually negligible.

In (1), the authors interpreted $Q(c)$ as approximating the probability by which the stochastic algorithm (e.g. SGD) outputs concept c after training. The preceding relaxes this assumption, because it shows that in some sense, the bound holds for "almost all" of the zero-error region of parameter space (where almost all should be interpreted as with high probability). A full analysis of SGD-trained neural networks still requires a rigorous analysis of SGD dynamics. However, we think the theorem strongly suggests our bounds to be applicable under a wide range of training schemes, as long as the realizability assumption is held (0 training error is reached).

2.1. Monte Carlo estimation of the kernel

The computational cost of computing the bound comes from computing $\log P(C(S))$, for which we use a Gaussian process approximation as in (1). For fully connected, convolutional and residual ReLU networks without pooling, we use previously published analytical forms of the kernel (16; 17).

For more complex architectures, we use Monte Carlo sampling as in (3). We find empirically that a sample size corresponding to a small constant times the size of the training set m works reasonably well[1].

3. Experiments

We performed a series of experiments to test the behaviour of the bound under a wide range of model/problem pairs. Here we refer as generalization error, to the empirical error on a test set, independent from the training set. This equals equals the true generalization error, with high probability, up to a negligible term.

In (1), the authors tested the bound on a fully connected and a convolutional network without pooling, on three datasets, as the amount of label corruption was increased (2). As can be seen in Fig. 1, the PAC-Bayes bounds closely follow the trends of decreasing generalization performance for increased corruption.

In Fig. 4, we compare the bounds and the true error for a series of modern architectures commonly used in computer vision tasks (See Appendix B for more details on the architectures). The networks are trained on a sample of 10k random images from CIFAR10, whose labels have been binarized (label 0 if class < 5, or 1 otherwise). The bound is non-vacuous for all the architectures, pretty tight for some architectures (densenet121), but more loose for others (resnet50, densenet201). It is encouraging to see the overall good performance of the bound. Nevertheless, the differences between performance for different architectures remains a subject of future investigation. One important factor may be related to the exponential growth of the kernel with depth, for some choices of weight/bias variance, as found in previous work (18; 16; 3). In Appendix C of (1), the effect of the variance parameters on the PAC-Bayes

[1]Note that at least m samples are needed if the empirical covariance matrix is to have full rank

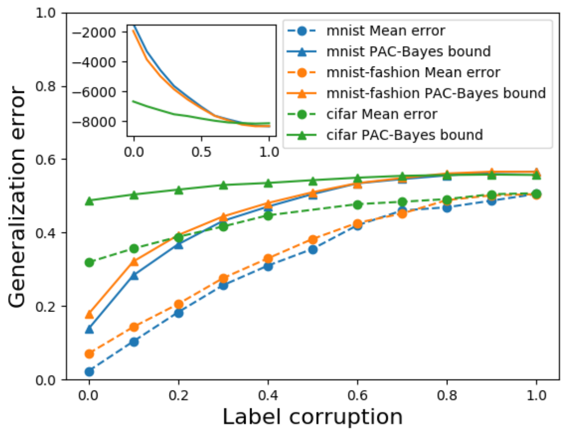

Figure 1. PAC-Bayes bound and generalization error versus training set size for three different datasets, for a CNN with 4 layers and no pooling, for three datasets and a training set of size 10000. Training set error is 0 in all experiments. The generalization errors are averaged over 8 initializations. The Gaussian process parameters were $\sigma_w = 1.0$, $\sigma_b = 1.0$. See Appendix B for more details on architecture. **Insets** show the marginal likelihood $P(C(S))$ of the data as computed by the Gaussian process approximation (in natural log scale), versus the label corruption. (reproduced with permission)

bound was also investigated.

An important practical question is how much generalization improves with increasing training data (which may be expensive to obtain). In Fig. 2, we plot the bound and the true test error as a function of the amount of training data m ranging from 1k to 40k, on the same datasets and architectures as in Fig. 1. The bound follows the same trend as the true error, and correctly captures the relative difficulty of the tasks (except for one point at $m = 1000$).

In Fig. 3, we show how the error and bound changes with the number of layers for a CNN with global max pooling on the last layer. We see that the bound follows the general downward trend of the true error as the number of layers increases.

4. Conclusion

In this work we have empirically tested the generalization error bound proposed in (1) on a variety of models and binary classification tasks, showing that the bound is nonvacuous and correlates with the true error on a wide range of situations model/problem pairs. We also proved a deterministic version of the PAC-Bayes bound, which further justifies the applicability of our results for deterministic networks trained with standard stochastic optimization algorithms.

Much of the discussion of generalization in neural networks

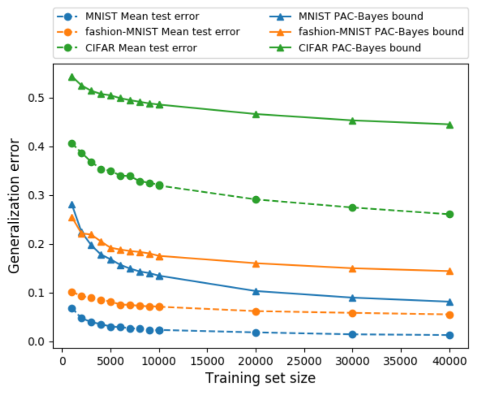

Figure 2. PAC-Bayes bound and generalization error versus training set size for three different datasets, for a CNN with 4 layers and no pooling. Training error is always 0.

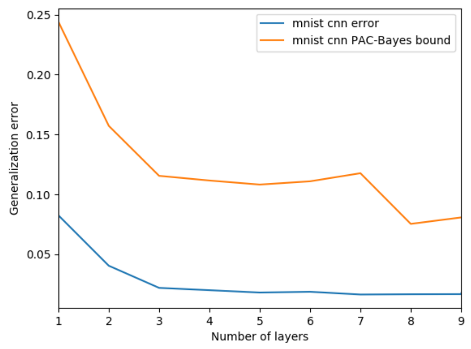

Figure 3. PAC-Bayes bound and generalization error versus number of layers for a CNN with max pooling trained on a sample of 10k MNIST images. Training error is always 0.

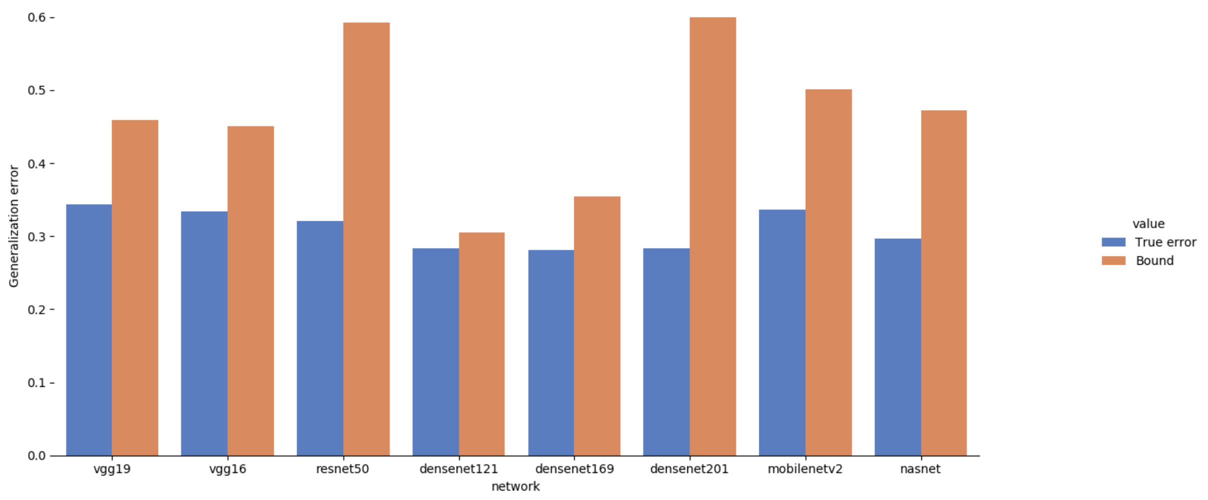

Figure 4. PAC-Bayes bound and generalization error for different architectures on a sample of 10k from CIFAR10 (with binary labels) as training set.

has focused on optimization methods (mainly SGD), and how optimizing hyperparameters can improve generalization. For practical applications such improvements can be extremely important. In this abstract, however, we focus on where the bulk of the generalization comes from, and not on further improvements. Nevertheless, it would be interesting to see if improvements due to optimization algorithm can also be rationalized in our PAC-Bayes formalism.

Remaining sources of error and areas of improvement within our PAC-Bayes formalism include:

- Finding tighter PAC-Bayesian bounds (improving on Theorem 1), and gain an understanding of optimal bounds under our set of assumptions (e.g. via lower bounds).
- Improved calculations of the marginal, e.g. by using MCMC algorithms, variational inference, or improved versions of the Expectation-Propagation / Laplacian approximations we use.
- Improving over the infinite layer width limit by a perturbation analysis of taking into account correlations, as is often done in physics to extend mean field theories.

References

[1] Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[2] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[3] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian convolutional neural networks with many channels are gaussian processes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[4] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.

[5] Yao Zhang, Andrew M Saxe, Madhu S Advani, and Alpha A Lee. Energy–entropy competition and the effectiveness of stochastic gradient descent in machine learning. *Molecular Physics*, pages 1–10, 2018.

[6] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[7] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

[8] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR, 10–15 Jul 2018.

[9] Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*, 2017.

[10] Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[11] Vaishnavh Nagarajan and Zico Kolter. Deterministic pac-bayesian generalization bounds for deep networks via generalizing noise-resilience. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[12] Gintare Karolina Dziugaite and Daniel M Roy. Data-dependent pac-bayes priors via differential privacy. *arXiv preprint arXiv:1802.09583*, 2018.

[13] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

[14] David A McAllester. Some pac-bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234. ACM, 1998.

[15] John Langford and Matthias Seeger. Bounds for averaging classifiers. 2001.

[16] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[17] Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow Gaussian processes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[18] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

A. Proof

Consider a concept c with generalization error $\epsilon(c)$. The probability that it has zero training error for a sample of m instances is $P\left[c \in C(S)\right] = (1-\epsilon(c))^{-m}$. This is smaller or equal to some $\delta > 0$ if

$$\ln\left(1-\epsilon(c)\right) \geq \frac{\ln\frac{1}{\delta}}{m}$$

This can be written as follows.

$$\forall c \forall \delta > 0 \forall^{\delta} S \left[c \in C(S) \text{ implies } \ln\left(1-\epsilon(c)\right) < \frac{\ln\frac{1}{\delta}}{m} \right]$$

By the quantifier reversal lemma (1),

$$\forall^{\delta} S \forall \alpha > 0 \forall^{\alpha} c \left[c \in C(S) \text{ implies } \ln\left(1-\epsilon(c)\right) < \frac{\ln\frac{1}{\alpha\beta\delta}}{(1-\beta)m} \right]$$

Let $B(S) \subseteq \mathcal{H}$ be the set of concepts violating the formula, then $P(B(S)) \leq \alpha$. Now, the conditional probability of a concept in $C(S)$ violating the formula is $\frac{P(B(S)\cap C(S))}{P(C(S))} \leq \frac{P(B(S))}{P(C(S))} \leq \frac{\alpha}{P(C(S))}$. This probability is therefore smaller than γ if $\alpha = \gamma P(C(S))$. We thus get

$$\forall^{\delta} S \forall \alpha > 0 \forall^{\gamma} c \in C(S) \left[\ln\left(1-\epsilon(c)\right) < \frac{\ln\frac{1}{P(C(S))} + \ln\frac{1}{\gamma\beta\delta}}{(1-\beta)m} \right]$$

We get the result, choosing $\beta = \frac{1}{m}$

$$\forall^{\delta} S \forall \alpha > 0 \forall^{\gamma} c \in C(S) \left[\ln\left(1-\epsilon(c)\right) < \frac{\ln\frac{1}{P(C(S))} + \ln m + \ln\frac{1}{\delta} + \ln\frac{1}{\gamma}}{m-1} \right]$$

B. Architecture details

In the main experiments of the paper we used two classes of architectures. Here we describe them in more detail.

- Fully connected networks (FCs), with varying number of layers. The size of the hidden layers was the same as the input dimension, and the nonlinearity was ReLU. The last layer was a single Softmax neuron. We used default Keras settings for initialization (Glorot uniform).
- Convolutional neural networks (CNNs), with varying number of layers. The number of filters was 200, and the nonlinearity was ReLU. The last layer was a fully connected single Softmax neuron. The filter sizes alternated between $(2, 2)$ and $(5, 5)$, and the padding between SAME and VALID, the strides were 1 (same default settings as in the code for (2)). We used default Keras settings for initialization (Glorot uniform).
- vgg16. Keras implementation (https://keras-contrib.readthedocs.io/en/latest/sources/applications/#vgg16). (3)
- vgg19. Keras implementation (https://keras-contrib.readthedocs.io/en/latest/sources/applications/#vgg19). (3)
- resnet50. Keras implementation (https://keras-contrib.readthedocs.io/en/latest/sources/applications/#resnet50). (4)
- densenet121, densenet169, densenet201. Keras implementation (https://keras-contrib.readthedocs.io/en/latest/sources/applications/#densenet). (5)

- mobilenetv2. Keras implementation (https://keras-contrib.readthedocs.io/en/latest/sources/applications/#mobilenet). (6)
- nasnet. Keras implementation (https://keras-contrib.readthedocs.io/en/latest/sources/applications/#nasnet). (7)

For vgg16,vgg19,resnet50,densenets, mobilenetv2, and nasnet, we used average global poolin in the last layer, unless otherwise specified.

References

[1] David A McAllester. Some pac-bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234. ACM, 1998.

[2] Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow Gaussian processes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[6] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[7] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.