Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data

Gintare Karolina Dziugaite

University of Cambridge

Daniel M. Roy

University of Toronto

Abstract. One of the defining properties of deep learning is that models are chosen to have many more parameters than available training data. In light of this capacity for overfitting, it is remarkable that simple algorithms like SGD reliably return solutions with low test error. One roadblock to explaining these phenomena in terms of implicit regularization, structural properties of the solution, and/or easiness of the data is that many learning bounds are quantitatively vacuous in this "deep learning" regime. In order to explain generalization, we need nonvacuous bounds. We return to an idea by Langford and Caruana (2001), who used PAC-Bayes bounds to compute nonvacuous numerical bounds on generalization error for *stochastic* two-layer two-hidden-unit neural networks via a sensitivity analysis. By optimizing the PAC-Bayes bound directly, we are able to extend their approach and obtain nonvacuous generalization bounds for deep stochastic neural network classifiers with millions of parameters trained on only tens of thousands of examples. We connect our findings to recent and old work on flat minima and MDL-based explanations of generalization.

Contents

1. Introduction

By optimizing a PAC-Bayes bound, we show that it is possible to compute nonvacuous numerical bounds on the generalization error of deep *stochastic* neural networks with millions of parameters, despite the training data sets being one or more orders of magnitude smaller than the number of parameters.

To our knowledge, these are the first explicit and nonvacuous numerical bounds computed in the deep learning regime where the number of network parameters eclipses the number of training examples. The bounds we compute are data dependent: indeed, even for relatively tiny neural networks, data independent bounds are vacuous (i.e., they bound the classification error by a number greater than 1). Evidently, we are operating far from the worst case.

Our investigation was instigated by recent empirical work by Zhang, Bengio, Hardt, Recht, and Vinyals [Zha+17], who show that stochastic gradient descent (SGD), applied to deep learning architectures with millions of parameters, is:

(1) able to achieve ≈ 0 training error on CIFAR10 and IMAGENET and still generalize (i.e., test error remains small, despite the potential for overfitting);
(2) still able to achieve ≈ 0 training error even after the labels are *randomized*, and does so with only a small factor of additional computational time.

Taken together, these two observations demonstrate that the network architecture has tremendous capacity to overfit and yet SGD does not abuse this capacity as it optimizes the surrogate loss, despite the lack of explicit regularization.

It is a major open problem to explain this phenomenon. A natural approach would be to show that, under natural conditions, SGD finds solutions that possess structural properties that we already know to be connected to generalization. However, in order to complete the logical connection, the associated learning bounds must be nonvacuous in the regime of model size / data size where we hope to explain the phenomenon.

This work establishes a potential candidate. Chernoff bounds on held-out data suggest our generalization bounds are loose: across a variety of network architectures, our PAC-Bayes bounds on the test error are in the range 16–22%, while Chernoff bounds on the test error based on held-out data are consistently around 3%. Despite the gap, theoreticians will likely be surprised that it is possible at all to obtain nonvacuous numerical bounds on generalization error for a model with such large capacity trained on so few training examples. While we cannot entirely explain the magnitude of generalization, we can demonstrate nontrivial generalization.

Our approach was inspired by a line of work in physics by Baldassi, Ingrosso, Lucibello, Saglietti, and Zecchina [Bal+15] and the same authors with Borgs and Chayes [Bal+16]. Based on theoretical results for discrete optimization linking computational efficiency to the existence of nonisolated solutions, the authors propose a number of new algorithms for learning discrete neural networks by explicitly driving them towards nonisolated solutions. On the basis of Bayesian ideas, they posit that these solutions should have good generalization properties. In a recent collaboration with Chaudhari, Choromanska, Soatto, and LeCun [Cha+17], they extend these ideas to modern deep learning architectures with continuous parametrizations, obtaining impressive empirical results.

In this setting, nonisolated solutions correspond to "flat minima". The existence and generalization properties of flat minima in the neural-network error surface is an old observation, going back at least to work by Hochreiter and Schmidhuber [HS97], who discuss sharp versus flat minima using the language of minimum description length (MDL; [Ris83]). In short, describing weights in sharp minima requires high precision in order to not incur nontrivial excess error, whereas flat minimum can be described with lower precision.

Hochreiter and Schmidhuber propose an algorithm to find flat minima by minimizing the training error while maximizing the log volume of a connected region of the parameter space that yields similar classifiers with similarly good training error.

There are very close connections—at both the level of analysis and algorithms—with the work of Chaudhari et al. [Cha+17] and close connections with the approach we take to compute nonvacuous generalization bounds by exploiting the local structure of the learned solution. (We discuss more related work in Section 6.)

Despite the promising theoretical underpinnings, the generalizations theorems given by [Cha+17] have admittedly unrealistic assumptions, and thus fall short of demonstrating that small local-entropic loss or the structure of the flat minimum found during optimization explains the generalization performance that they observe.

The goal of this work is to identify structure in the solutions obtained by SGD that provably implies small generalization error. Computationally, it is much easier to demonstrate that a randomized classifier will generalize, and so our results actually pertain to the generalization error of a *stochastic* neural network, i.e., one whose weights/biases are drawn at random from some distribution on every forward evaluation of the network. In order to explain the observations of [Zha+17], the next step would be to study whether such structure necessarily arises from performing SGD under natural conditions. (We suspect one condition may be that the Bayes error rate is close to zero.) More ambitiously, perhaps the same structure can explain also the efficiency of SGD in practice.

1.1. **Approach.** Our working hypothesis is that SGD finds good solutions only if they are surrounded by a relatively large volume of solutions that are nearly as good. This hypothesis suggests that PAC-Bayes bounds may be fruitful: if SGD finds a solution contained in a large volume of equally good solutions, then the expected error rate of a classifier drawn at random from this volume should match that of the SGD solution. The PAC-Bayes theorem [McA99] bounds the expected error rate of a classifier chosen from a distribution Q in terms of the Kullback–Liebler divergence from some a priori fixed distribution P, and so if the volume of equally good solutions is large, and not too far from the mass of P, we will obtain a nonvacuous bound.

Our approach will be to use optimization to find a broad distribution Q over neural network parameters that minimizes the PAC-Bayes bound, in effect mapping out the volume of equally good solutions surrounding the SGD solution. This idea is actually a modern take on an old idea by Langford and Caruana [LC02], who apply PAC-Bayes bounds to small two-layer stochastic neural networks (with only 2 hidden units) that were trained on (relatively large, in comparison) data sets of several hundred labeled examples.

The basic idea can be traced back even further to work by Hinton and Camp [HC93], who propose an algorithm for controlling overfitting in neural networks via the minimum description length principle. In particular, they minimize the sum of the empirical squared error and the KL divergence between a prior and posterior distribution on the weights. Their algorithm is applied to networks with 100's of inputs and 4 hidden units, trained on several hundred labeled examples. Hinton and Camp do not compute numerical generalization bounds to verify that MDL principles alone suffice to *explain* the observed generalization.

Our algorithm more directly extends the work by Langford and Caruana, who propose to construct a distribution Q over neural networks by performing a sensitivity analysis on each parameter after training, searching for the largest deviation that does not increase the training error by more than, e.g., 1%. For Q, Langford and Caruana choose a multivariate normal distribution over the network parameters, centered at the parameters of the trained neural network. The covariance matrix is diagonal, with the variance of each parameter chosen to be the estimated sensitivity, scaled by a global constant. (The global scale is chosen so that the

training error of Q is within, e.g., 1% of that of the original trained network.) Their prior P is also a multivariate Gaussian, but with zero mean and covariance given by some scalar multiple of the identity matrix. By employing a union bound, they allow themselves to choose the scalar multiple in a data-dependent fashion to optimize the PAC-Bayes bound.

The algorithm sketched by Langford and Caruana does not scale to modern neural networks for several reasons, but one dominates: in massively overparametrized networks, individual parameters often have negligible effect on the training classification error, and so it is not possible to estimate the *relative* sensitivity of large populations of neurons by studying the sensitivity of neurons in isolation.

Instead, we use stochastic gradient descent to directly optimize the PAC-Bayes bound on the error rate of a stochastic neural network. At each step, we update the network weights and their variances by taking a step along an unbiased estimate of the gradient of (an upper bound on) the PAC-Bayes bound. In effect, the objective function is the sum of i) the empirical surrogate loss averaged over a random perturbation of the SGD solution, and ii) a generalization error bound that acts like a regularizer.

Having demonstrated that this simple approach can construct a witness to generalization, it is worthwhile asking whether these ideas can be extended the setting of local-entropic loss [Cha+17]. If we view the distribution that defines the local-entropic loss as defining a stochastic neural network, can we use PAC-Bayes bounds to establish nonvacuous bounds on its generalization error?

2. Preliminaries

Much of the setup is identical to that of [LC02]: We are working in the batch supervised learning setting. Data points are elements $x \in \mathcal{X} \subseteq \mathbb{R}^k$ with binary class labels $y \in \{-1, 1\}$. We will denote the training set of size m as S_m:

$$S_m = \{(x_i, y_i)\}_{i=1,\dots,m}, \text{ where } (x_i, y_i) \in (\mathcal{X} \times \mathcal{Y}). \tag{1}$$

Let $\mathcal{M}$ be all probability measures on the data space $\mathbb{R}^k \times \{-1, 1\}$. We will assume that the training examples are i.i.d. samples from some $\mu \in \mathcal{M}$.

A parametric family of classifiers is a function $H : \mathbb{R}^d \times \mathbb{R}^k \to \{-1, 1\}$, where $h_w := H(w, \cdot) : \mathbb{R}^k \to \{-1, 1\}$ is the classifier indexed by the parameter $w \in \mathbb{R}^d$. The hypotheses space induced by H is $\mathcal{H} = \{h_w : w \in \mathbb{R}^d\}$. A randomized classifier is a distribution Q on $\mathbb{R}^d$. Informally, we will speak of distributions on $\mathcal{H}$ when we mean distributions on the underlying parametrization.

We are interested in the 0–1 loss $\ell : \mathbb{R} \times \{-1, 1\} \to \{0, 1\}$

$$\ell(\hat{y}, y) = \mathbb{I}(\text{sign}(\hat{y}) = y). \tag{2}$$

We will also make use of the logistic loss $\breve{\ell} : \mathbb{R} \times \{-1, 1\} \to \mathbb{R}_+$

$$\breve{\ell}(\hat{y}, y) = \frac{1}{\log(2)} \log\big(1 + \exp(-\hat{y}y)\big), \tag{3}$$

which will serve as a convex surrogate (i.e., upper bound) to the 0–1 loss.

We define the following notions of error:

- $\hat{e}(h, S_m) = \dfrac{1}{m} \sum_{i=1}^{m} \ell(h(x_i), y_i)$ empirical classification error of hypothesis h for sample S_m;
- $\breve{e}(h, S_m) = \dfrac{1}{m} \sum_{i=1}^{m} \breve{\ell}(h(x_i), y_i)$ empirical (surrogate) error of a hypothesis h on the training data set S_m. We will use this for training purposes when we need our empirical loss to be differentiable;

- $e_\mu(h) = \mathop{\mathbb{E}}_{S_m \sim \mu^m}[\hat{e}(h, S_m)]$ expected error for hypothesis h under the data distribution μ (we will often drop the subscript μ and just write $e(h)$);
- $\hat{e}(Q, S_m) = \mathop{\mathbb{E}}_{w \sim Q}[\hat{e}(h_w, S_m)]$ expected empirical error under the randomized classifier Q on $\mathcal{H}$;
- $e(Q) = \mathop{\mathbb{E}}_{w \sim Q}[e_\mu(h_w)]$ expected error for Q on $\mathcal{H}$.

2.1. **KL divergence.** Let Q, P be probability measures defined on a common measurable space $\mathcal{H}$, such that Q is absolutely continuous with respect to P, and write $\frac{\mathrm{d}Q}{\mathrm{d}P} : \mathcal{H} \to \mathbb{R}_+ \cup \{\infty\}$ for some Radon–Nikodym derivative of Q with respect to P. Then the Kullback–Liebler divergence (or relative entropy) of P from Q is defined to be

$$\mathrm{KL}(Q||P) := \int \log \frac{\mathrm{d}Q}{\mathrm{d}P} \, \mathrm{d}Q. \tag{4}$$

We will mostly be concerned with KL divergences where Q and P are probability measures on Euclidean space, $\mathbb{R}^d$, absolutely continuous with respect to Lebesgue measure. Let q and p denote the respective densities. In this case, the definition of the KL divergence simplifies to

$$\mathrm{KL}(Q||P) = \int \log \frac{q(x)}{p(x)} q(x) \mathrm{d}x. \tag{5}$$

Of particular interest to us is the KL divergence between multivariate Gaussian distributions in $\mathbb{R}^d$. Let $N_q = \mathcal{N}(\mu_q, \Sigma_q)$ be a multivariate Gaussian with mean μ_q and covariance matrix Σ_q, let $N_p = \mathcal{N}(\mu_p, \Sigma_p)$, and assume Σ_q and Σ_p are positive definite. Then

$$\mathrm{KL}(N_q||N_p) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_p^{-1}\Sigma_q\right) - k + (\mu_p - \mu_q)^\top \Sigma_p^{-1}(\mu_p - \mu_q) + \ln\left(\frac{\det \Sigma_p}{\det \Sigma_q}\right)\right). \tag{6}$$

For $p, q \in [0, 1]$, we will abuse notation and define

$$\mathrm{KL}(q||p) := \mathrm{KL}(\mathcal{B}(q)||\mathcal{B}(p)) = q \log \frac{q}{p} + (1 - q) \log \frac{1 - q}{1 - p}, \tag{7}$$

where $\mathcal{B}(p)$ denotes the Bernoulli distribution on $\{0, 1\}$ with mean p.

2.2. **Inverting KL bounds.** In the following sections, we will encounter bounds of the form

$$\mathrm{KL}(q||p) \leq c \tag{8}$$

for $q, p \in [0, 1]$ and $c \geq 0$. We will most often be interested in the quantity

$$\mathrm{KL}^{-1}(q|c) := \sup\{p \in [0, 1] \, : \, \mathrm{KL}(q||p) \leq c\}. \tag{9}$$

We are not aware of a simple formula for $\mathrm{KL}^{-1}(q|c)$, although numerical approximations are readily obtained via Newton's method (Appendix B). For the purpose of gradient-based optimization, we can use the well-known inequality, $2(q - p)^2 \leq \mathrm{KL}(q||p)$, to obtain a simple upper bound

$$\mathrm{KL}^{-1}(q|c) \leq q + \sqrt{c/2}, \tag{10}$$

which holds whenever the right hand side is bounded by 1. This bound is quantitatively loose when $q \approx 0$, because then $\mathrm{KL}^{-1}(q|c) \approx c$ for $c \ll 1$, as compared with the upper bound of $\Theta(\sqrt{c})$. On the other hand, when c is large enough that $q + \sqrt{\frac{c}{2}} > 1$, the derivative of $\mathrm{KL}^{-1}(q|c)$ is zero, whereas the upper bound provides a useful derivative.

2.3. **Bounds.** We will employ three bounds to control the generalization error: the union bound, a sample convergence bound derived from the Chernoff bound, and the PAC-Bayes bound due to McAllester [McA99]. We state the union bound for completeness.

Theorem 2.1 (union)**.** *Let $E_1, E_2, \ldots$ be events. Then $\mathbb{P}(\bigcup_n E_n) \leq \sum_n \mathbb{P}(E_n)$.*

Recall that $\mathcal{B}(p)$ denotes the Bernoulli distribution on $\{0,1\}$ with mean $p \in [0,1]$. The following bound is derived from the KL formulation of the Chernoff bound:

Theorem 2.2 (sample convergence [LC02])**.** *For every $p, \delta \in (0,1)$ and $n \in \mathbb{N}$,*

$$\mathop{\mathbb{P}}_{x \sim \mathcal{B}(p)^n} \left(\mathrm{KL}(n^{-1} \textstyle\sum_{i=1}^n x_i || p) \geq \frac{\log \frac{2}{\delta}}{n} \right) \leq \delta. \tag{11}$$

Finally, we present a variant of the PAC-Bayes bound due to Langford and Seeger [LS01]. (See also [Lan02].) The PAC-Bayes theorem was first established by McAllester [McA99].

Theorem 2.3 (PAC-Bayes [McA99; LS01])**.** *For every $\delta > 0$, $m \in \mathbb{N}$, distribution μ on $\mathbb{R}^k \times \{-1, 1\}$, and distribution P on $\mathcal{H}$,*

$$\mathop{\mathbb{P}}_{S_m \sim \mu^m} \left((\exists Q)\ \mathrm{KL}(\hat{e}(Q, S_m) || e(Q)) \geq \frac{\mathrm{KL}(Q||P) + \log \frac{m}{\delta}}{m-1} \right) \leq \delta. \tag{12}$$

The PAC-Bayes bound leads to the following learning algorithm [McA99]:

(1) Fix a probability $\delta > 0$ and a distribution P on $\mathcal{H}$.
(2) Collect an i.i.d. dataset S_m of size m.
(3) Compute the distribution Q on $\mathcal{H}$ that minimizes the error bound

$$\mathrm{KL}^{-1}\left(\hat{e}(Q, S) \middle| \frac{\mathrm{KL}(Q||P) + \log \frac{m}{\delta}}{m-1} \right). \tag{13}$$

(4) Return the randomized classifier corresponding to Q.

In all but the simplest scenarios, this learning algorithm is intractable. However, we can attempt to approximate it.

3. Optimizing the PAC-Bayes bound

Let H be a parametric family of classifiers and write h_w for $H(w, \cdot)$. We will interpret h_w as a neural network with (weight/bias) parameters $w \in \mathbb{R}^d$, although the development below is more general.

Fix $\delta \in (0,1)$ and some distribution P on $\mathbb{R}^d$, and let $S_m \sim \mu^m$ be m i.i.d. training examples. We are interested in minimizing the PAC-Bayes bound Eq. (13) with respect to Q.

For every $w \in \mathbb{R}^d$ and $s \in \mathbb{R}^d_+$, let $\mathcal{N}_{w,s} = \mathcal{N}(w, \mathrm{diag}(s))$ denote the multivariate normal distribution with mean w and diagonal covariance $\mathrm{diag}(s)$. As our first simplifications, we replace the PAC-Bayes with the upper bound described by Eq. (10), replace the empirical loss with its convex surrogate, and restrict Q to the family of multivariate normal distributions with diagonal covariance structure, yielding the optimization problem

$$\min_{w \in \mathbb{R}^d, s \in \mathbb{R}^d_+} \quad \breve{e}(\mathcal{N}_{w,s}, S_m) + \sqrt{\frac{\mathrm{KL}(\mathcal{N}_{w,s}||P) + \log \frac{m}{\delta}}{2(m-1)}}. \tag{14}$$

3.1. **The Prior.** It remains to choose the prior P. Choosing P to be multivariate normal leads to a simple analytical formula for the KL divergence. Symmetry considerations would suggest that we choose $P = \mathcal{N}(0, \lambda I)$ for some $\lambda > 0$, however there is no single good choice of λ. (We will also see that there are good reasons not to choose a zero mean, and so we will let w_0 denote the mean to be chosen a priori.)

In order to deal with the problem of choosing λ, we will follow Langford and Caruana [LC02] and use a Structural Risk Minimization type argument to choose λ optimally from a discrete set, at the cost of a slight expansion to our generalization bound. In particular, we will take $\lambda = c\exp\{-j/b\}$ for some $j \in \mathbb{N}$ and fixed $b, c \geq 0$. If the PAC-Bayes bound for each $j \in \mathbb{N}$ is designed to hold with probability at least $1 - \frac{6}{\pi^2 j^2}$, then, by the union bound (Theorem 2.1), it will hold uniformly for all $j \in \mathbb{N}$ with probability at least $1 - \delta$, as desired.[1] During optimization, we will want to avoid discrete optimization, and so we will treat λ as if it were a continuous variable. (We will then discretize λ when we evaluate the PAC-Bayes bound after the fact.) Solving for j, we have $j = b\log\frac{c}{\lambda}$, and so we will replace j with this term during optimization. Taking into account the choice of P and the continuous approximation to the union bound, we have the following minimization problem:

$$\min_{w\in\mathbb{R}^d, s\in\mathbb{R}^d_+, \lambda\in(0,c)} \quad \breve{e}(\mathcal{N}_{w,s}, S_m) + \sqrt{B_{\mathrm{RE}}(w, s, \lambda; \delta)} \tag{15}$$

where

$$B_{\mathrm{RE}}(w, s, \lambda; \delta) = \frac{\mathrm{KL}(\mathcal{N}_{w,s}||\mathcal{N}(w_0, \lambda I)) + 2\log(b\log\frac{c}{\lambda}) + \log\frac{\pi^2 m}{6\delta}}{2(m-1)} \tag{16}$$

and, using Eq. (6), the KL term simplifies to

$$\mathrm{KL}(N_{w,s}||N(w_0, \lambda I)) = \frac{1}{2}(\frac{1}{\lambda}\|s\|_1 + \frac{1}{\lambda}\|w - w_0\|_2^2 + d\log\lambda - 1_d \cdot \log s - d). \tag{17}$$

3.2. **Stochastic Gradient Descent.** We cannot optimize this objective directly because we cannot compute $\breve{e}(\mathcal{N}_{w,s}, S_m)$ or its gradients efficiently. We can, however, employ a type of stochastic gradient descent. Therefore, the final simplification that we will make is to take gradient steps with respect to the unbiased estimator $\breve{e}(h_{w'}, S_m)$, $w' \sim \mathcal{N}_{w,s}$. We will use a new independent unbiased estimate at each iteration. Note that we compute the estimate of the gradient with respect to the entire data set, although one could also have used a random mini-batch of the training data at each step. Given the stochastic approximation to the error term, we can now employ any gradient-based optimization algorithm.

3.3. **Evaluating the final PAC-Bayes bound.** While we treat λ as a continuous parameter during optimization, the union bound requires that λ be of the form $\lambda = c\exp\{-j/b\}$, for some $j \in \mathbb{N}$. We therefore round λ up or down, choosing that which delivers the best bound, as computed below.

According to the PAC-Bayes and union bound, with probability $1-\delta$, uniformly over all $w \in \mathbb{R}^d$, $s \in \mathbb{R}^d_+$, and λ of the form $c\exp\{-j/b\}$, for $j \in \mathbb{N}$, the error rate of the randomized classifier $Q = \mathcal{N}_{w,s}$ is bounded by

$$\mathrm{KL}^{-1}(\hat{e}(Q, S)|B_{\mathrm{RE}}(w, s, \lambda; \delta)). \tag{18}$$

We cannot compute this bound exactly because computing $\hat{e}(Q, S)$ is intractable. However, we can obtain unbiased estimates and apply the sample convergence

[1] A superior but intractable approach is to choose P as a scale mixture of multivariate normal distributions. If the scale mixture is chosen to "match" the prior we have defined here, then we would expect the bound to be tighter because nearby values of j cannot "share statistical strength", while they would were P a mixture.

bound (Theorem 2.2). In particular, given n i.i.d. samples $w_1, \ldots, w_n$ from Q, we produce the Monte Carlo approximation $\hat{Q}_n = \sum_{i=1}^n \delta_{w_i}$, for which $\hat{e}(\hat{Q}_n, S_m)$ is exactly computable, and obtain the bound

$$\hat{e}(Q, S) \leq \overline{\hat{e}_{n,\delta'}}(Q, S) := \mathrm{KL}^{-1}(\hat{e}(\hat{Q}_n, S_m) | n^{-1} \log 2/\delta'), \tag{19}$$

which holds with probability $1 - \delta'$. By another application of the union bound,

$$e(Q) \leq \mathrm{KL}^{-1}(\overline{\hat{e}_{n,\delta'}}(Q, S) | B_{\mathrm{RE}}(w, s, \lambda; \delta)), \tag{20}$$

with probability $1 - \delta - \delta'$. We use this bound in our reported results.

4. Experiments

We optimize PAC-Bayes bounds on the error rates of stochastic neural networks trained on a binary classification variant of MNIST. We train several different network architectures, varying both the depth and the width of the network. We obtain nonvacuous generalization bounds for networks with large VC dimension.

4.1. **Dataset.** We use the MNIST handwritten digits data set [LCB10] as provided in Tensorflow [Aba+15], where the dataset is split into the training set (55000 images) and test set (10000 images). (We do not use the validation set.) Each MNIST image is black and white and 28-pixels square, resulting in a network input dimension of $k = 784$. MNIST is usually treated as a multiclass classification problem. In order to use standard PAC-Bayes bounds, we produce a binary classification problem by mapping numbers $\{0, \ldots, 4\}$ to label 1 and $\{5, \ldots, 9\}$ to label -1. In some experiments, we train on random labels, i.e., binary labels drawn independently and uniformly at random.

4.2. **Initial network training using SGD.** All experiments are performed on fully connected feed-forward neural networks with 2–4 layers. We choose a standard initialization scheme for the weights and biases: Weights are initialized randomly from a normal distribution (with mean zero and standard deviation 0.04) that is truncated to [-0.08, 0.08]. Biases are initialized to a constant value of 0.1 for the first layer and 0 for the remaining layers.

We use REctified Linear Unit (RELU) activations at every hidden node. The last layer is linear. We minimize the logistic loss by Stochastic Gradient Descent (SGD) and Momentum: learning rate 0.01; momentum 0.9. SGD is run in mini-batches of size 100.

On our binary variant of MNIST, we train several neural network architectures of varying depth and width (see Table 1). In each case, we train for a total of 20 epochs. We also train a small network with 1 hidden layer of 600 nodes on *random* labels, in order to demonstrate the large capacity of the network. Obtaining $\approx$0 training error required 120 epochs. See the first two rows of Table 1 for the train/test error rates.

4.3. **PAC-Bayes bound optimization.** As described in Section 3, we optimize $w \in \mathbb{R}^d$, $s \in \mathbb{R}^d_+$, and $\lambda \in (0, c)$ according to Eq. (15) using gradient descent, replacing the empirical surrogate error of the randomized classifier $Q = \mathcal{N}_{w,s}$ with an unbiased estimate produced from a single sample from Q at each iteration. Before optimization, we fix $\delta = 0.025$, $b = 100$, and $c = 0.1$.

To ensure that the variables $\lambda \in (0, c)$ and $s \in \mathbb{R}^d_+$ remain positive, we reparametrize and instead optimize variables representing $\frac{1}{2}\log(\lambda)$ and $\frac{1}{2}\log(s)$.

We run gradient descent with the RMSprop optimizer (decay 0.9). The learning rate is set to 0.001 for the first 150000 iterations. We then lower it to 0.0001 for the final 50000 iterations. For the random label experiment, we optimize the bound with a smaller learning rate 0.0001 for 500000 iterations.

Algorithm 1 is pseudo code for optimizing the PAC-Bayes bound. The code implements vanilla SGD, although it can be easily modified to use other optimizer, e.g., RMSprop, momentum.

4.4. **Initialization and Symmetry Breaking.** In an ideal world, we would account for all the network symmetries when computing the KL divergence in the PAC-Bayes bound. (See Appendix A for a discussion.) Because it does not seem to be computational feasible to account for the symmetries, it makes sense to try to break the symmetries somehow. In fact, one consequence of randomly initializing a network is that some symmetries are broken. If we do not expect SGD to reverse these symmetries, then the initial weight configuration, which we will denote by w_0, will be a better mean for the prior P than the origin. In fact, breaking the symmetries in this way lead to much better bounds than setting the means to zero.

The prior variance λ is initialized to a fixed valued of $\exp\{-6\}$, and the sensitivities s to $|w|$ with $a = 1$. For experiments with random labels, we take s to be $|w|/10$.

4.5. **Reported values.** All reported error rates correspond to classification error.

The train and test errors are evaluated on the network learned by SGD. In all experiments, SGD achieves perfect or near-perfect classification accuracy on the training data. We start from the SGD solution when optimizing the PAC-Bayes bound.

The reported SNN train and test error rates are upper bounds computed by an application of Theorem 2.2 as described in Section 3.3 with $\delta' = 0.01$ and $n = 150000$. These numbers are chosen in order to get the estimate within 0.001–0.002. The reported test error of the SGD solution is the empirical mean. (In light of 10000 test data points and the observed error rates, upper bounds via Theorem 2.2 are only 0.005 higher.)

The PAC-Bayes bound is computed as described in Section 3.3. Each bound holds with probability 0.965 over the choice of the training set and the draws from the learned SNN Q. For the random label experiment, we report $\sqrt{B_{\mathrm{RE}}(w, s, \lambda; \delta)}$ as defined in Eq. (16), since the PAC-Bayes bound cannot be computed for values greater than 1.

To calculate the upper bound on the VC dimension of the network, we use an upper bound communicated to us by Bartlett [Bar17] which is itself in $O(LW \log W)$, where L is the number of layers and W is the total number of tunable parameters.

5. Results

See Table 1. All SGD trained networks achieve perfect or nearly perfect accuracy on the training data. On true labels, the SNN mean training error increases slightly as the weight distribution broadens to minimize the KL divergence. The SGD solution is close to mean of the SNN as measured with respect to the SNN covariance. For the random label experiment, the SNN mean training error rises above 10%. Ideally, it might have risen to nearly 50%, while driving down the KL term to near zero.

The empirical test error of the SGD classifiers does not change much across the different architectures, despite the potential for overfitting. This phenomenon is well known, though still remarkable. For the random label experiment, the empirical test classification error of 0.508 represents lack of generalization, as expected. The same two patterns hold for the SNN test error too, with slightly higher error rates.

Remarkably, the PAC-Bayes bounds do not grow much despite the networks becoming several times larger, and all true label experiments have classification

error bounded by 0.23. Since larger networks possess many more symmetries, the true PAC-Bayes bounds for our learned stochastic neural network classifier might be substantially smaller. (See Appendix A for a discussion.) While these bounds are several times larger than the test error estimated on held out data (approximately, 0.03), they demonstrate nontrivial generalization. As expected mathematically, the PAC-Bayes bound on the classification error for random labels is 1.0, indicating the absence of generalization.

The VC dimension upper bounds indicate that data independent bounds will be vacuous by several orders of magnitude. Because the number of parameters exceeds the available training data, lower bounds imply that generalization cannot be explained in a data independent way.

Experiment	T-600	T-1200	T-300^2	T-600^2	T-1200^2	T-600^3	R-600
Train error	0.001	0.002	0.000	0.000	0.000	0.000	0.007
Test error	0.018	0.018	0.015	0.016	0.015	0.013	0.508
SNN train error	0.028	0.027	0.027	0.028	0.029	0.027	0.112
SNN test error	0.034	0.035	0.034	0.033	0.035	0.032	0.503
PAC-Bayes bound	0.161	0.179	0.170	0.186	0.223	0.201	1.352
KL divergence	5144	5977	5791	6534	8558	7861	201131
# parameters	471601	943201	326101	832201	2384401	1192801	471601
VC dimension	26m	56m	26m	66m	187m	121m	26m

TABLE 1. Results for experiments on binary classification variant of MNIST dataset. SGD is either trained on (T) true labels or (R) random labels. The network architecture is expressed as N^L, indicating L hidden layers with N nodes each. Errors are classification error. The reported VC dimension is the best known upper bound (in millions) for ReLU networks. The SNN error rates are tight upper bounds (see text for details). The PAC-Bayes bounds upper bound the test error with probability 0.965.

5.1. **Parameter optimization.** One question we were interested in was whether the weights obtained from optimizing the PAC-Bayes bound had changed much from the SGD solution w_{SGD} that served as an initialization. To answer this question, we calculated the p-value of the SGD solution under the distribution of the stochastic neural network.

Let Q_{SNN} denote the distribution obtained by optimizing the PAC-Bayes bound, write w_{SNN} and Σ_{SNN} for its mean and covariance, and let $\|w\|_{\Sigma_{\mathrm{SNN}}} = w^T \Sigma_{\mathrm{SNN}}^{-1} w$ denote the induced norm. Using 10000 samples, we estimated

$$\mathop{\mathbb{P}}_{w\sim Q_{\mathrm{SNN}}}\Big(\|w - w_{\mathrm{SNN}}\|_{\Sigma_{\mathrm{SNN}}} < \|w_{\mathrm{SGD}} - w_{\mathrm{SNN}}\|_{\Sigma_{\mathrm{SNN}}}\Big). \tag{21}$$

The estimate was 0 for all true label experiments, i.e., w_{SGD} is less extreme of a perturbation of w_{SNN} than a typical perturbation. For the random label experiments, w_{SNN} and w_{SGD} differ significantly, which is consistent with the bound being optimized in the face of random labels.

6. Related work

As we mention in the introduction, our approach scales the ideas in [HC93] and [LC02] to the modern deep learning regime where the networks have millions of parameters, but are trained on one or two orders of magnitude fewer training examples. The objective we optimize is an upper bound on the PAC-Bayes bound, which we know from the discussion in Section 2.2 will be very loose when the

Algorithm 1 PAC-Bayes bound optimization by SGD

Input:

$w_0 \in \mathbb{R}^d$ ▷ Network parameters (random init.)
$w \in \mathbb{R}^d$ ▷ Network parameters (SGD solution)
S_m ▷ Training examples
$\delta \in (0,1)$ ▷ Confidence parameter
$b \in \mathbb{N}, c \in (0,1)$ ▷ Precision and bound for λ
$\tau \in (0,1), T$ ▷ Learning rate; # of iterations

Output: Optimal w, s, λ ▷ Weights, variances

procedure PAC-BAYES-SGD
  $\varsigma \leftarrow \mathrm{abs}(w)$ ▷ where $\varsigma = \log\sqrt{s}$
  $\varrho \leftarrow -3$ ▷ where $\varrho = \log\sqrt{\lambda}$
  $\mathrm{B}(w, s, \lambda, w') = \breve{e}(h_{w'}, S_m) + \sqrt{\frac{1}{2} B_{\mathrm{RE}}(w, s, \lambda)}$
  for $t \leftarrow 1, T$ **do** ▷ Run SGD for T iterations.
    Sample $\tilde{w} \sim \mathcal{N}(w, \mathrm{diag}(\mathrm{e}^{2\varsigma}))$
    ▷ Gradient step

$$\begin{bmatrix} w \\ \varsigma \\ \varrho \end{bmatrix} = \begin{bmatrix} w \\ \varsigma \\ \varrho \end{bmatrix} - \tau \begin{bmatrix} \nabla_w B(w, \mathrm{e}^{2\varsigma}, \mathrm{e}^{2\varrho}, \tilde{w}) \\ \nabla_\varsigma B(w, \mathrm{e}^{2\varsigma}, \mathrm{e}^{2\varrho}, \tilde{w}) \\ \nabla_\varrho B(w, \mathrm{e}^{2\varsigma}, \mathrm{e}^{2\varrho}, \tilde{w}) \end{bmatrix}$$

  return $w, \mathrm{e}^{2\varsigma}, \mathrm{e}^{2\varrho}$

empirical classification error is approximately zero. Indeed, in that case, the PAC-Bayes bound is approximately

$$\hat{e}(\mathcal{N}_{w,s}, S_m) + \frac{\mathrm{KL}(\mathcal{N}_{w,s}||P) + \log\frac{m}{\delta}}{(m-1)}. \tag{22}$$

The objective optimized by Hinton and Camp is of the same essential form as this one, except for the choice of squared error and different prior and posterior distributions. We explored using Eq. (22) as our objective with a surrogate loss, but it did not produce better results.

In the introduction we discuss the close connection of our work to several recent papers [Bal+15; Bal+16; Cha+17] that study "flat" or nonisolated minima on the account of their generalization and/or algorithmic properties.

Based on theoretical results for k-SAT that efficient algorithms find nonisolated solutions, Baldassi et al. [Bal+16] model efficient neural network learning algorithms as minimizers of a *replicated* version of the empirical loss surface, which emphasizes nonisolated minima and deemphasizes isolated minima. They then propose several algorithms for learning discrete neural networks using these ideas.

In follow-up work with Chaudhari, Choromanska, Soatto, and LeCun [Cha+17], they translate these ideas into the setting of continuously parametrized neural networks. They introduce an algorithm, called Entropy-SGD, which seeks out large regions of dense local minima: it maximizes the depth and flatness of the energy landscape. Their objective integrates both the energy of nearby parameters and the weighted distance to the parameters. In particular, rather than directly minimizing an error surface $w \mapsto L(h_w, S_m)$, they propose the following minimization problem over the so-called local-entropic loss:

$$\min_{w \in \mathbb{R}^d} \quad \log \mathop{\mathbb{E}}_{W \sim \mathcal{N}_{w,1/\gamma}} [C(\gamma) \exp\{-L(h_W, S_m)\}], \tag{23}$$

where $\gamma > 0$ is a parameter and $C(\gamma)$ a constant. In comparison, our algorithm can be interpreted as an optimization of the form

$$\min_{w \in \mathbb{R}^p, s \in \mathbb{R}^p_+} \quad \mathop{\mathbb{E}}_{W \sim \mathcal{N}_{w,s}} [L(h_W, S_m)] \; + \; R(w, s) \tag{24}$$

where R serves as a regularizer that accounts for the generalization error by, roughly speaking, trying to expand the axis-aligned ellipsoid $\{x \in \mathbb{R}^d : (w - x)^T \mathrm{diag}(s)^{-1}(w - x) = 1\}$ and draw it closer to some point w_0 near the origin. Comparing Eqs. (23) and (24) highlights similarities and differences. The local-entropic loss is sensitive to the volume of the regions containing good solutions. While the first term in our objective function looks similar, it does not, on its own, account for the volume of regions. This role is played by the second term, which prefers large regions (but also ones near the initialization w_0). In our formulation, the first term is the empirical error of a stochastic neural network, which is precisely the term whose generalization error we are trying to bound. Entropy-SGD was not designed for the purpose of finding good stochastic neural networks, although it seems possible that having small local-entropic loss would lead to generalization for neural networks whose parameters are drawn from the local Gibbs distribution. Another difference is that, in our formulation, the shape of the Gaussian perturbation is learned adaptively, and driven by the goal of minimizing generalization error. The shape of the Gaussian perturbation is not learned, although the region whose volume is being measured is determined by the error surface, and it seems likely that this volume will be larger than that spanned by a multivariate Gaussian chosen to lie entirely in a region with good loss.

Chaudhari et al. [Cha+17] give an informal characterization of the generalization properties of local-entropic loss in Bayesian terms by comparing the marginal likelihood of two Bayesian priors centered at a solution with small and large local-entropic loss. Informally, a Bayesian prior centered on an isolated solution will lead to small marginal likelihood in contrast to one centered in a wide valley. They give a formal result relying on the uniform stability of SGD [HRS15] to show under some strong (and admittedly unrealistic) conditions that Entropy-SGD generalizes better than SGD. The key property is that the local-entropic loss surface is smoother than the original error surface.

Other authors have found evidence of the importance of "flat" minima: Recent work by Keskar, Mudigere, Nocedal, Smelyanskiy, and Tang [Kes+17] finds that large-batch methods tend to converge to sharp / isolated minima and have worse generalization performance compared to mini-batch algorithms, which tend to converge to flat minima and have good generalization performance. The bulk of their paper is devoted to the problem of restoring good generalization behavior to batch algorithms.

Dinh, Pascanu, Bengio, and Bengio [Din+17] criticize various notions of "flatness" and argue that flatness, as defined, is not a necessary condition for generalization. They do this by demonstrating reparameterizations that relocate "flat" minima to yield sharp minima. However, one can logically explain the generalization properties of SGD solutions in terms of sufficient conditions, including the "flatness" of minima or volume of nearly equivalent solutions.

Finally, our algorithm also bears resemblance to *graduated optimization*, an approach toward non-convex optimization attributed to Blake and Zisserman [BZ87] whereby a sequence of increasingly fine-grained versions of an optimization problem are solved in succession. (See [HLS16] and references therein.) In this context, Eq. (23) is the result of a local smoothing operation acting on the objective function $w \mapsto \breve{\ell}(h_w, S_M)$. In graduate optimization, the effect of the local smoothing operation would be decreased over time, eventually disappearing. In our formulation,

the act of balancing the empirical loss and generalization error serve to drive the evolution of the local smoothing in an adaptive fashion. Moreover, in the limit, the local smoothing does not vanish in our algorithm, as the volume spanned by the perturbations relates to the generalization error. Our results suggest that SGD solutions live inside relatively large volumes, and so perhaps SGD can be understood in terms of graduated optimization.

7. Conclusions and Future work

We obtain nonvacuous generalization bounds for deep neural networks with millions of parameters trained on 55000 MNIST examples. These bounds are obtained by optimizing an objective derived from the PAC-Bayes bound, starting from the solution produced by SGD. Despite the weights changing, the SGD solution remains well within the 1% ellipsoidal quantile, i.e., the volume spanned by the stochastic neural network contains the original SGD solution. (When labels are randomized, however, optimizing the PAC-Bayes bound causes the solution to shift considerably.)

Our experiments look only at fully connected feed forward networks trained on a binary classification problem derived from MNIST. It would be interesting to see if the results extend to multiclass classification, to other data sets, and to other types of architectures, especially convolutional ones.

In our experiments, we optimize the PAC-Bayes bound starting from an SGD solution. One could instead train the network entirely by optimizing the PAC-Bayes bound from a random initialization and study how the optima and bounds compare to those produced via SGD. There are other changes worth studying: highly dependent weights constrain the size of the axis-aligned ellipsoid representing the stochastic neural network. We can potentially recognize small populations of highly dependent weights, and optimize their covariance parameters, rather than enforcing independence in the posterior.

It is also interesting to consider replacing the posterior with a distribution that is more tuned to the loss surface. One promising avenue is to follow the lines of Chaudhari et al. [Cha+17] and consider (local) Gibbs distributions. If the solutions obtained by minimizing the local-entropic loss are flatter than those obtained by SGD, than we may be able to demonstrate quantitatively tighter bounds.

Finally, there is the hard work of understanding the generalization properties of SGD. In light of our work, it may be useful to start by asking whether SGD finds solutions in flat minima. Such solutions could then be lifted to stochastic neural networks with good generalization properties. Going from stochastic networks back to deterministic ones may require additional structure, such as margin.

Acknowledgments

This research was carried out while the authors were visiting the Simons Institute for the Theory of Computing at UC Berkeley. The authors would like to thank Peter Bartlett, Shai Ben-David, Dylan Foster, Matus Telgarsky, and Ruth Urner for helpful discussions. GKD is supported by an EPSRC studentship. DMR is supported by an NSERC Discovery Grant, Connaught Award, and U.S. Air Force Office of Scientific Research grant #FA9550-15-1-0074.

References

[Aba+15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[Bal+15] C. Baldassi, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina. "Subdominant Dense Clusters Allow for Simple Learning and High Computational Performance in Neural Networks with Discrete Synapses". *Phys. Rev. Lett.* 115 (12 Sept. 2015), p. 128101.

[Bal+16] C. Baldassi, C. Borgs, J. T. Chayes, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina. "Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes". *Proceedings of the National Academy of Sciences* 113.48 (2016), E7655–E7662. eprint: http://www.pnas.org/content/113/48/E7655.full.pdf.

[Bar17] P. L. Bartlett. "The impact of the nonlinearity on the VC-dimension of a deep network". Preprint. 2017.

[BZ87] A. Blake and A. Zisserman. *Visual Reconstruction*. Cambridge, MA, USA: MIT Press, 1987.

[Cha+17] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. "Entropy-SGD: Biasing Gradient Descent Into Wide Valleys". In: *International Conference on Learning Representations (ICLR)*. 2017. arXiv: 1611.01838v4 [cs.LG].

[Din+17] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. *Sharp Minima Can Generalize For Deep Nets*. 2017. arXiv: 1703.04933v1 [cs.LG].

[HC93] G. E. Hinton and D. van Camp. "Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights". In: *Proceedings of the Sixth Annual Conference on Computational Learning Theory*. COLT '93. Santa Cruz, California, USA: ACM, 1993, pp. 5–13.

[HLS16] E. Hazan, K. Y. Levy, and S. Shalev-Shwartz. "On Graduated Optimization for Stochastic Non-Convex Problems". In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2016, pp. 1833–1841.

[HRS15] M. Hardt, B. Recht, and Y. Singer. "Train faster, generalize better: Stability of stochastic gradient descent". *CoRR* abs/1509.01240 (2015).

[HS97] S. Hochreiter and J. Schmidhuber. "Flat Minima". *Neural Comput.* 9.1 (Jan. 1997), pp. 1–42.

[Kes+17] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *International Conference on Learning Representations (ICLR)*. 2017. arXiv: 1609.04836v2 [cs.LG].

[Lan02] J. Langford. "Quantitatively tight sample complexity bounds". Carnegie Mellon University, 2002.

[LC02] J. Langford and R. Caruana. "(Not) Bounding the True Error". In: *Advances in Neural Information Processing Systems 14*. Ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, 2002, pp. 809–816.

[LCB10] Y. LeCun, C. Cortes, and C. J. C. Burges. *MNIST handwritten digit database*. http://yann.lecun.com/exdb/mnist/. 2010.

[LS01] J. Langford and M. Seeger. *Bounds for Averaging Classifiers*. Tech. rep. CMU-CS-01-102. Carnegie Mellon University, 2001.

[McA99] D. A. McAllester. "PAC-Bayesian Model Averaging". In: *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*. COLT '99. Santa Cruz, California, USA: ACM, 1999, pp. 164–170.

[Ris83] J. Rissanen. "A Universal Prior for Integers and Estimation by Minimum Description Length". *Ann. Statist.* 11.2 (June 1983), pp. 416–431.

[Zha+17] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. "Understanding deep learning requires rethinking generalization". In: *International Conference on Representation Learning (ICLR)*. 2017. arXiv: 1611.03530v2 [cs.LG].

A. Network symmetries

Fix a neural network architecture $H : \mathbb{R}^d \times \mathbb{R}^k \to \{-1, 1\}$ and write h_w for $H(w, \cdot)$. It has long been appreciated that distinct parametrizations $w, w' \in \mathbb{R}^d$ can lead to the same *functions* $h_w = h_{w'}$, and so the set $\mathcal{H} = \{h_w : w \in \mathbb{R}^d\}$ of classifiers defined by a neural network architecture is a quotient space of $\mathbb{R}^d$.

For the purposes of understanding the generalization error of neural networks, we would ideally work directly with $\mathcal{H}$. Let P, Q be a distributions on $\mathbb{R}^d$, i.e., stochastic neural networks. Then P and Q induce distributions on $\mathcal{H}$, which we will denote by $\bar{P}$ and $\bar{Q}$, respectively. For the purposes of the PAC-Bayes bound, it is the KL divergence $\mathrm{KL}(\bar{Q}||\bar{P})$ that upper bounds the performance of the stochastic neural network Q. In general, $\mathrm{KL}(\bar{Q}||\bar{P}) \leq \mathrm{KL}(Q||P)$, but it is difficult in practice to approximate the former because the quotient space is extremely complex.

One potential way to approach $\mathcal{H}$ is to account for symmetries in the parameterization. A *network symmetry* is a map $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ such that, for all $w \in \mathbb{R}^d$, we have $h_w = h_{\sigma(w)}$. As an example of such a symmetry, in a fully connected network with identical activation functions at every unit, the function computed by the network is invariant to permuting the nodes with a hidden layer. Let $\mathbb{S}$ be any finite set of symmetries possessed by the architecture. For every distribution Q on $\mathbb{R}^d$ and network symmetry σ, we may define $Q_\sigma = Q \circ \sigma^{-1}$ to be the distribution over networks obtained by first sampling network parameters from Q and then applying the map σ to obtain a network that computes the same function.

Define $Q^{\mathbb{S}} = \frac{1}{|\mathbb{S}|} \sum_{\sigma \in \mathbb{S}} Q_\sigma$. Informally, Q and $Q^{\mathbb{S}}$ are identical when viewed as distributions on functions, yet $Q^{\mathbb{S}}$ spreads its mass evenly over equivalent parametrizations. In particular, for any data set S, we have $\hat{e}(Q, S) = \hat{e}(Q^{\mathbb{S}}, S)$. We call $Q^{\mathbb{S}}$ a symmetrized version of Q. The following lemma states that symmetrized versions always have smaller KL divergence with respect to distributions that are invariant to symmetrization: Before stating the lemma, recall that the differential entropy of an absolutely continuous distribution Q on $\mathbb{R}^d$ with density q is $\int q(x) \log q(x) \mathrm{d}x \in \mathbb{R} \cup \{-\infty, \infty\}$.

Lemma A.1. *Let $\mathbb{S}$ be a finite set of network symmetries, let P be an absolutely continuous distribution such that $P = P_\sigma$ for all $\sigma \in \mathbb{S}$, and define $Q^{\mathbb{S}}$ as above for some arbitrary absolutely continuous distribution Q on $\mathbb{R}^d$ with finite differential entropy. Then* $\mathrm{KL}(Q^{\mathbb{S}}||P) = \mathrm{KL}(Q||P) - \mathrm{KL}(Q||Q^{\mathbb{S}}) \leq \mathrm{KL}(Q||P)$.

The above lemma can be generalized to distributions over (potentially infinite) sets of network symmetries.

It follows from this lemma that one can do no worse by accounting for symmetries using mixtures, provided that one is comparing to a distribution P that is invariant to those symmetries. In light of the PAC-Bayes theorem, this means that a generalization bound based upon a KL divergence that does not account for symmetries can likely be improved. However, for a finite set $\mathbb{S}$ of symmetries, it is easy to show that the improvement is bounded by $\log|\mathbb{S}|$, which suggests that, in order to obtain appreciable improvements in a numerical bound, one would need to account for an exponential number of symmetries. Unfortunately, exploiting this many symmetries seems intractable. It is hard to obtain useful lower bounds to $\mathrm{KL}(Q||Q^{\mathbb{S}})$, while upper bounds from Jensen's inequality led us to negative (hence vacuous) lower bounds on $\mathrm{KL}(Q^{\mathbb{S}}||P)$.

In this work, we therefore take a different approach to dealing with symmetries. Neural networks are randomly initialized in order to *break* symmetries. Combined with the idea that the learned parameters will reflect these broken symmetries, we choose our prior P to be located at the random initialization, rather than at zero.

B. Approximating $\mathrm{KL}^{-1}(q|c)$

There is no simple formula for $\mathrm{KL}^{-1}(q|c)$, but we can approximate it via root-finding techniques. For all $q \in (0,1)$ and $c \geq 0$, define $h_{q,c}(p) = \mathrm{KL}(q||p) - c$. Then $h'_{q,c}(p) = \frac{1-q}{1-p} - \frac{q}{p}$. Given a sufficiently good initial estimate p_0 of a root of $h_{q,c}(\cdot)$, we can obtain improved estimates of a root via Newton's method:

$$p_{n+1} = \mathrm{N}(p_n; q, c) \text{ where } \mathrm{N}(p; q, c) = p - \frac{h_{q,c}(c)}{h'_{q,c}(p)}. \tag{25}$$

This suggests the following approximation to $\mathrm{KL}^{-1}(q|c)$:

(1) Let $\tilde{b} = q + \sqrt{\frac{c}{2}}$.
(2) If $\tilde{b} \geq 1$, then return 1.
(3) Otherwise, return $\mathrm{N}^k(\tilde{b})$, for some small integer $k > 0$.

Our reported results use five steps of Newton's method.