CBMM Memo No. 067 **July 19, 2017**

Theory of Deep Learning III: Generalization Properties of SGD

by

Chiyuan Zhang[1] Qianli Liao[1] Alexander Rakhlin[2] Brando Miranda[1] Noah Golowich[1] Tomaso Poggio[1]

[1]Center for Brains, Minds, and Machines, McGovern Institute for Brain Research,
Massachusetts Institute of Technology, Cambridge, MA, 02139.
[2] University of Pennsylvania

Abstract: In Theory III we characterize with a mix of theory and experiments the consistency and generalization properties of deep convolutional networks trained with Stochastic Gradient Descent in classification tasks. A present perceived puzzle is that deep networks show good predicitve performance when the classical learning theory seems to suggest overfitting. We describe an explanation of these empirical results in terms of the following new results on SGD:

1. SGD concentrates in probability - like the classical Langevin equation – on large volume, "flat" minima, selecting flat minimizers which are also global minimizers.
2. Minimization under the constraint of maximum volume (usually corresponding to flatness) yields through the jacobian wrt weights and the jacobian wrt x, large (geometrical) margin classification in the case of separable data (zero empirical error for classification loss).
3. Large geometrical margin implies classification bounds via robustness theorems. These bounds can qualitatively explain all the generalization properties empirically observed for deep networks.

Thus SGD selects minimizers corresponding to maximum geometrical margin. Within a single flat minimum the average of the asymptotic fluctuations for each of the degenerate directions (and the non-degenerate ones) is at the maximum margin (the variance however is expected to increase with time in the presence of noise). Because of its connection with robust optimization, SGD can be shown to perform a form of implicit regularization. This explains the puzzling findings about fitting randomly labeled data while performing well on natural labeled data. It also explains while overparametrization does not result in overfitting. Quantitative, non-vacuous bounds are still missing as it has almost always been the case for most practical applications of machine learning. We describe in the appendix an alternative approach that explains with tools of linear algebra the same qualitative properties and puzzles of generalization in deep polynomial networks.

This is version 2. The first version was released on 04/04/2017 at https://dspace.mit.edu/handle/1721.1/107841.

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF - 123 1216.

1 Introduction

In the last few years, deep learning has been tremendously successful in many important applications of machine learning. However, our understanding of deep learning is still far from complete. A satisfactory characterization of deep learning should at least cover the following parts: 1) representation power — what types of functions can neural networks (DNNs) represent well and what are the advantages and disadvatages over using shallow models? 2) optimization of the empirical loss — can we characterize the convergence of stochastic gradient descent (SGD) on the non-convex empirical loss encountered in deep learning? 3) why do the deep learning models, despite being highly over-parameterized, still predict well?

The first two questions are addressed in Theory I [1] and Theory II [2] respectively. In this paper, we focus on the last question, and try to provide a hybrid theoretical and empirical view of it. More specifically, we study learning, especially in over-parameterized DNNs, by optimizing the empirical loss with SGD.

We remind that *generalization* is defined as the following property: the gap between empirical and expected error goes to zero with increasing size of the training set. *Consistency* is a different property: the empirical error converges to the optimal Bayes error with increasing size of the training set. Both properties can be distribution dependent or independent and both have strong and weak forms.

In the case of *one-pass* SGD, where each training point is only visited at most once, the algorithm is optimizing the expected loss directly. Therefore, there is no need to define the empirical loss and to measure generalization: consistency is key here and it holds under rather general conditions. However, in practice, unless one has access to infinite training samples, one-pass SGD is rarely used. Instead, it is almost always better to run many passes of SGD over the same training set. In this case, the algorithm is optimizing the empirical loss, and the deviation between the empirical loss and the expected loss (i.e. the generalization error) must be controlled.

In statistical learning theory, the deviation is usually controlled by restricting the complexity of the hypothesis space. For example, in binary classification, for a hypothesis space with VC-dimension d and N i.i.d. training samples, the generalization error could be upper bounded by $\mathcal{O}(\sqrt{d/N})$. In the *distribution-free setting*, the VC dimension also provide a lower bound for the generalization error. On the other hand, for overparametrization ($d \gg N$), there is a data distribution under which the generalization error could be arbitrarily bad [3]. As we will discuss later this worst case behavior was recently demonstrated by a randomization test on large deep neural networks that have the full capability of shattering the whole training set [4]. In those experiments, zero-error minimizers for the empirical loss are found by SGD. Since the test performance must be at chance level, the worst possible generalization error is observed. On the other hand, those same networks are found to achieve low expected error on image classification datasets, without obvious regularization-like mechanisms such as weight decay or data augmentation. This creates an apparent puzzle that we will discuss in section 6.1 as the classical characterization of distribution-independent generalization no longer readily applies. We observe that the puzzle is misleading because consistency may hold in the absence of generalization. A very classical example is 1-Nearest Neighbor, which is an algorithm with zero empirical error for all N and good asymptotic performance in terms of expected error. More in general, k-NN algorithms for all fixed k do not have generalization. In fact, if the Bayes error is zero, k-NN algorithms are consistent for all k!

In this paper, we speak loosely about two regimes for training deep convolutional networks (in the specific case of CIFAR). In the underconstrained regime – which is the standard one for deep learning applications – in which $n \leq W$, where n is the size of the training set and W is the number of weights– the empirical error can be zero, when, as described in Theory I and II, the regression function underlying the problem is contained in the class of (compositional) functions that are well approximated by the convolutional network and the weights in the network are sufficient for it (this exclude too many "dummy" weights) . On the other hand, the generalization error - the difference between empirical and expected error – may not go to zero for $n \to \infty$. We show that the expected error in the underconstrained regime should be low because SGD implicitely maximizes (geometrical) margin. In the over-constrained regime – $n \geq W$ – generalization follows from classical bounds on neural networks with a finite number of weights. As side-effects of these basic results we show close relations between SGDL, robust optimization wrt perturbations of the weights, large geometrical margin and regularization. Finally, we discuss how recent puzzling results on generalization by deep (and shallow) networks can be explained by this theory.

In the rest of the paper, we try to address this set of issues at the level of formal rigor of a physicist (not a mathematician: this will come later). With a mix of theoretical and empirical results, we show that isotropic flatness around the global minimizers play a key role in the generalization of over-parameterized deep neural networks.

Notice that in all computer simulations reported in this paper, we turn off all the "tricks" used to improve performance such as data augmentation, weight decay etc. in order to study the basic properties of the SGD algorithm. As a consequence, performance is not state of the art but maximum performance is not our goal here.

1.1 Related work

Deep Learning references start with Hinton's back-propagation and with LeCun's convolutional networks (see [5] for a nice review). Of course, multi-layer convolutional networks have been around at least as far back as the optical processing era of the 70s. The Neocognitron[6] was a convolutional neural network that was trained to recognize characters, inspired by the hierarchical processing postulated by Hubel and Wiesel[7] from their recordings of simple and complex cells in visual cortex. The property of *compositionality* was a main motivation for hierarchical models of visual cortex such as HMAX which was described as a pyramid of AND and OR layers[8], that is a sequence of conjunctions and disjunctions. In Theory I [1] we have provided formal conditions under which deep networks can use the compositionality to avoid the curse of dimensionality.

A number of recent papers have mentioned some of the ideas explored in this paper. For instance Soatto et al. [9] remark that almost-flat regions of the energy landscape are robust to data perturbations, noise in the activations as well as perturbations of the parameters — which are as we show later directly related to good generalization. They also note that wide valleys should result in better generalization and that optimization algorithms in deep learning seem to discover exactly that.

Recent work by Keskar et al.[10] is even more relevant. The authors estimate the loss in a neighborhood of the weights to argue that small batch size in SGD (i.e., larger gradient estimation noise, see later) generalizes better than large mini-batches and also results in significantly flatter minima. In particular, they note that the stochastic gradient descent method used to train deep nets, operate in a small-batch regime wherein a fraction of the training data, usually between 32 and 512 data points, is sampled to compute an approximation to the gradient. They discuss the common observation that when using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize. We provide theoretical arguments for the cause for this generalization drop in the large-batch regime, supporting the numerical evidence of Keskar et al.. The latter shows that large-batch methods tend to converge to sharp minimizers of the training and testing functions — and that sharp minima lead to poor generalization. In contrast, small-batch methods consistently converge to minimizers that generalize better, and our theoretical arguments support a commonly held view that this is due to the inherent noise in the gradient estimation.

On the other hand, as shown in [11], sharp minimizers do not necessarily lead to bad generalization performance. Due to the parameterization redundancy in deep neural networks, given any (flat) minimizers, one can artificially transform the weights to land in a sharp but equivalent minimizer because the function defined by the weights are the same. However, the argument in [11] does not conflict with our argument that flat minimizers generalize well. Moreover, it is not clear whether SGD will select the somewhat artificial sharp minimizers created by the equivalent transformations. Notice that isotropic flat minima in the loss wrt *all weights* – in which we are interested – cannot be transformed in sharp minima.

Another recent paper (after the first version of this memo) [12] is relevant in terms of the intuition about margin and low expected error (but not generalization as claimed!).

The puzzles mentioned in the abstract and text refer mainly to the recent work of one of us [4], which is one of the motivations of this paper. Even more recently, [13] descibe similar puzzles for kernel machines trained without regularization terms by SGD. Our results should be helpful in explaining them.

2 Background Facts

We describe here several previous results and observations that are key for our results about generalization.

2.1 Landscape of the Empirical Risk

In Theory II [2] we have described some features of the landscape of the empirical risk, for the case of deep networks of the compositional type (with weight sharing, though the proofs do not need the weight sharing assumption). We assumed over-parametrization, that is more parameters than training data points, as most of the successful deep networks. Under these conditions, setting the empirical error to zero yields a system of ϵ-approximating polynomial equations that have an infinite number of solutions (for the network weights). Alternatively, one can replace the RELUs in the network with an approximating univariate polynomial and verify empirically that the network behavior is basically unchanged. The associated system of equations allows for a large number of solutions – when is not inconsistent – which are *degenerate, that is flat* in several of the dimensions (in CIFAR there are about 10^6 unknown parameters for 6×10^4 equations. Notice that solutions with zero empirical error are global minimizers. No other solution with zero-error exists with a deeper minimum or less generic degeneracy. Empirically we observe (see Theory II) that zero-minimizers correspond to isotropically flat regions – not just valleys. In this paper, we will use the word flatness in two distict meanings (the context makes clear which meaning): one to refer to degeneracy of the minimizers, the other to refer to isotropical flatness around the zero of the empirical loss.

2.2 SGD: Basic Setting

Let Z be a probability space with an unknown measure ρ. A training set S_n is a set of i.i.d. samples $z_i, i = 1, \dots, n$ from ρ. Assume a hypothesis $\mathcal{H}$ is chosen in advance of training. Here we assume $\mathcal{H}$ is a p-dimensional Hilbert space, and identify elements of $\mathcal{H}$ with p-dimensional vectors in $\mathbb{R}^p$. A loss function is a map $V : \mathcal{H} \times Z \to \mathbb{R}_+$. Moreover, we assume the expected loss

$$I(f) = \mathbb{E}_z V(f, z) \tag{1}$$

exists for all $f \in \mathcal{H}$. We consider the problem of finding a minimizer of $I(f)$ in a closed subset $K \subset \mathcal{H}$. We denote this minimizer by f_K so that

$$I(f_K) = \min_{f \in K} I(f) \tag{2}$$

In general, the existence and uniqueness of a minimizer is not guaranteed unless some further assumptions are specified.

Since ρ is unknown, we are not able evaluate $I(f)$. Instead, we try to minimize the empirical loss

$$I_{S_n}(f) = \hat{\mathbb{E}}_{z \sim S_n} V(f, z) = \frac{1}{n} \sum_{i=1}^{n} V(f, z_i) \tag{3}$$

as a proxy. In deep learning, the most commonly used algorithm is SGD and its variants. The basic version of SGD is defined by the following iterations:

$$f_{t+1} = \Pi_K \left(f_t - \gamma_t \nabla V(f_t, z_t) \right) \tag{4}$$

where z_t is a sampled from the training set S_n uniformly at random, and $\nabla V(f_t, z_t)$ is an unbiased estimator of the full gradient of the empirical loss at f_t:

$$\mathbb{E}_{z_t \sim S_n} \left[\nabla V(f_t, z_t) \right] = \nabla \hat{I}(f_t)$$

γ_t is a decreasing sequence of non-negative numbers, usually called the *learning rates* or *step sizes*. $\Pi_K : \mathcal{H} \to K$ is the projection map into K, and when $K = \mathcal{H}$, it becomes the identity map. It is interesting that the following equation, labeled SGDL, and studied by several authors, including [14], seem to work as well as or better than the usual repeat SGD used to train deep networks, as discussed in Section 5:

$$f_{t+1} = f_t - \gamma_n \nabla V(f_t, z_t) + \gamma_t' W_t. \tag{5}$$

Here W_t is a standard Gaussian vector in $\mathbb{R}^p$ and γ_t' is a sequence going to zero.

We consider a situation in which the expected cost function $I(f)$ can have multiple *global* minima. As argued by [15] there are two ways to prove convergence of SGD. The first method consists of partitioning the parameter space into several attraction basins, assume that after a few iterations the algorithm confines the parameters in a single attraction basin, and proceed as in the convex case. A simpler method, instead of proving that the function f converges, proves that the cost function $I(f)$ and its gradient $\mathbb{E}_z \nabla V(f, z))$ converge.

Existing results show that when the learning rates decrease with an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to a global minimum when the objective function is convex or pseudoconvex[1], and otherwise converges almost surely to a local minimum. This direct optimization shortcuts the usual discussion for batch ERM about differences between optimizing the empirical risk on S_n and the expected risk.

Often extra-assumptions are made to ensure convergence and generalization by SGD. Here we observe that simulations on standard databases remain essentially unchanged if the domain of the weights is assumed to be a torus which is compact, because the weights remain bounded in most cases.

3 Consistency and Generalization

" Folk theorems" as well as classical bounds (see 8.1) suggest that generalization requires more data than the number W of "effective" parameters – though of course the number of parameters is not a general measure of the network complexity that is relevant for generalization. This is typically not the case in the current use of deep networks. Curiously, the recent wave of papers related to prediction performance of deep learning has mostly neglected the important fact that good accuracy on new data (consistency) can take place in the absence of generalization – defined as the convergence of empirical risk to expected risk as the number of data N increases. As we mentioned already, a classical example is the NN algorithm which can be proven to perform quite well in general but does not generalize in a distribution-independent way.

Loosely speaking we distinguish two phases (in principle both n and W grow to infinity):

1. $n \geq W$ where N is the number of training data and W is the number of "effective" parameters: this is the "classical" situation for generalization
2. $n \leq W$: this is the "new" regime in which SGD performs well in terms of consistency but may not have generalization.

Figure 1 shows that at least for this experiment on CIFAR there seem to be generalization for $N > W$ as expected and, though there is no generalization for $N < W$, the test error – as a proxy for the expected error – is good. The random label case in Figure 2 makes it very clear. The expected error is at chance level throughout; the training error is zero for $n << W$, but starts to increase to reach the expected error for $n > W$. The behavior in the "classical" regime – on the right side of the figures – can be accounted for by a several theoretical approaches that we describe in this section. The behavior in the second regime requires a novel (but see Appendix 9.10) set of arguments (see next section).

The classical theory proves characterizes generalization behavior as a function of n the number of training examples. But a more practical question is which architecture to choose for a fixed, given number of examples N. In fact, Figures 1 and 2 prompt some additional questions:

[1] In convex analysis, a pseudoconvex function is a function that behaves like a convex function with respect to finding its local minima, but need not actually be convex. Informally, a differentiable function is pseudoconvex if it is increasing in any direction where it has a positive directional derivative.

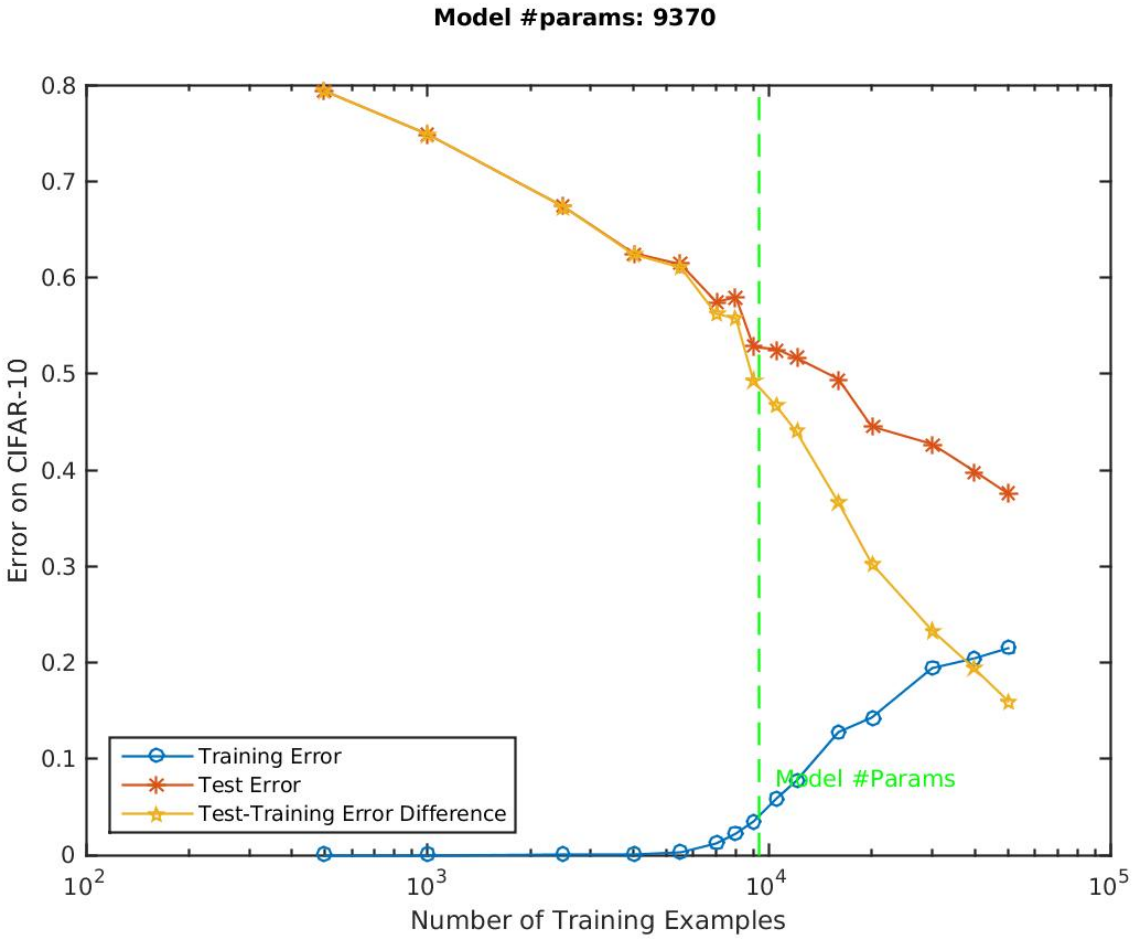


Figure 1: The figure shows the behavior of a deep network trained on subsets of the CIFAR database. The network is a 5-layer all convolutional network (i.e., no pooling) with 16 channels per hidden layer , resulting in only $W \approx 10000$ weights instead of the typical $300,000$. Neither data augmentation nor regularization is performed. The two regimes described in the text are for $N > W$ and $N < W$ respectively.

- for a fixed N is it better wrt expected error to train a deep network with $N \geq W$ or with $N \leq W$?

- is the mechanism explaining generalization the same that explain good expected error performance for $N \leq W$?

The empirical answer to these questions is shown in Figures 3 and 4 and it is surprising: it appears that the increase in the number of parameters does not yield any overfitting. We will first deal with the classical regime $N \ll W$ and then take on the challenge of explaining the non-overfitting behavior shown on the right of Figure 3.

4 Classical generalization bounds

This section though necessary for completeness is too boring. As a consequence, it landed in the Appendix which the reader may consult because some of the definitions necessary for the following are there. The Appendix reviews a number of approaches used to prove generalization – that is convergence to zero of the gap between empirical and expected error for $n \to \infty$. Though some of them do not explicitely require $n \gg W$, they "seem" empirically vacuous in our experiments with CIFAR for $n \ll W$.

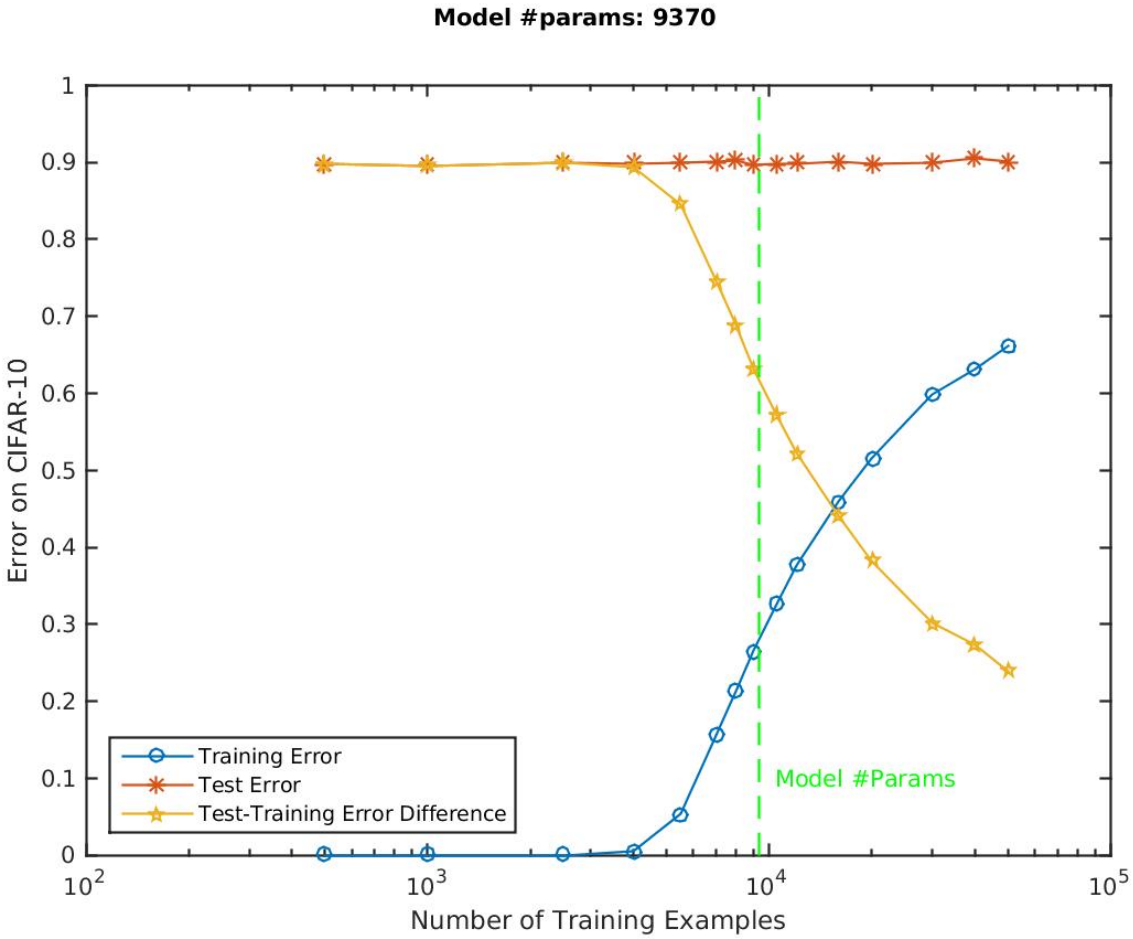


Figure 2: The figure shows the behavior of a deep network trained on subsets of the CIFAR database in which the labels have been randomly scrambled. The network is a 5-layer all convolutional network (i.e., no pooling) with 16 channels per hidden layer, resulting in only $W \approx 10000$ weights instead of the typical $300,000$. Neither data augmentation nor regularization is performed. The two regimes described in the text are for $N > W$ and $N < W$ respectively.

5 The puzzle: SGD for the underdetermined case

The puzzle is outside classical learning theory: why there is no overfitting for increasing overparametrization? Why is the expected error so low in the non-classical regime of $N < W$ in which we cannot speak about generalization? Our conjecture is that

Conjecture 1.
- *SGD, while minimizing the empirical loss also maximizes the volume, that is "flatness", of the minima, that is robustness to perturbations of the effective weights*
- *"flatness" is equivalent to large geometrical margin*
- *large geometrical margin by itself implies low expectation error for classification*

Our argument can be loosely described as follows. In the overparametrized regime, SGD selects with high probability large volume minima, that – apart from pathological cases – correspond to flat minima, because those correspond to large regions of zero loss. Notice that at a flat minimum, the pseudonoise of SGD is effectively switched off in expectation (!) because the expected gradient is zero. Large volume, flat minima correspond to large margin solutions: the intuition is that they are robust to perturbations of the weights and thus to corresponding perturbations of the input vector. The last step in the argument is that large margin solutions generalize well in the case of classification. Thus increasing the number of parameters does not change this regularizing behavior of SGD, exactly as in the linear case. As a side rmeark, we will later show that SGD is similar to robust optimization, providing implicit regularization.

A similar argument underlies the following intuition that links Theory II with this memo: the unique zero minimizers for $n < W$ but close to W become degenerate, that is flat, for $n \leq W$. So each set of degenerate zeros for $n \leq W$ can be thought as a class of equivalence

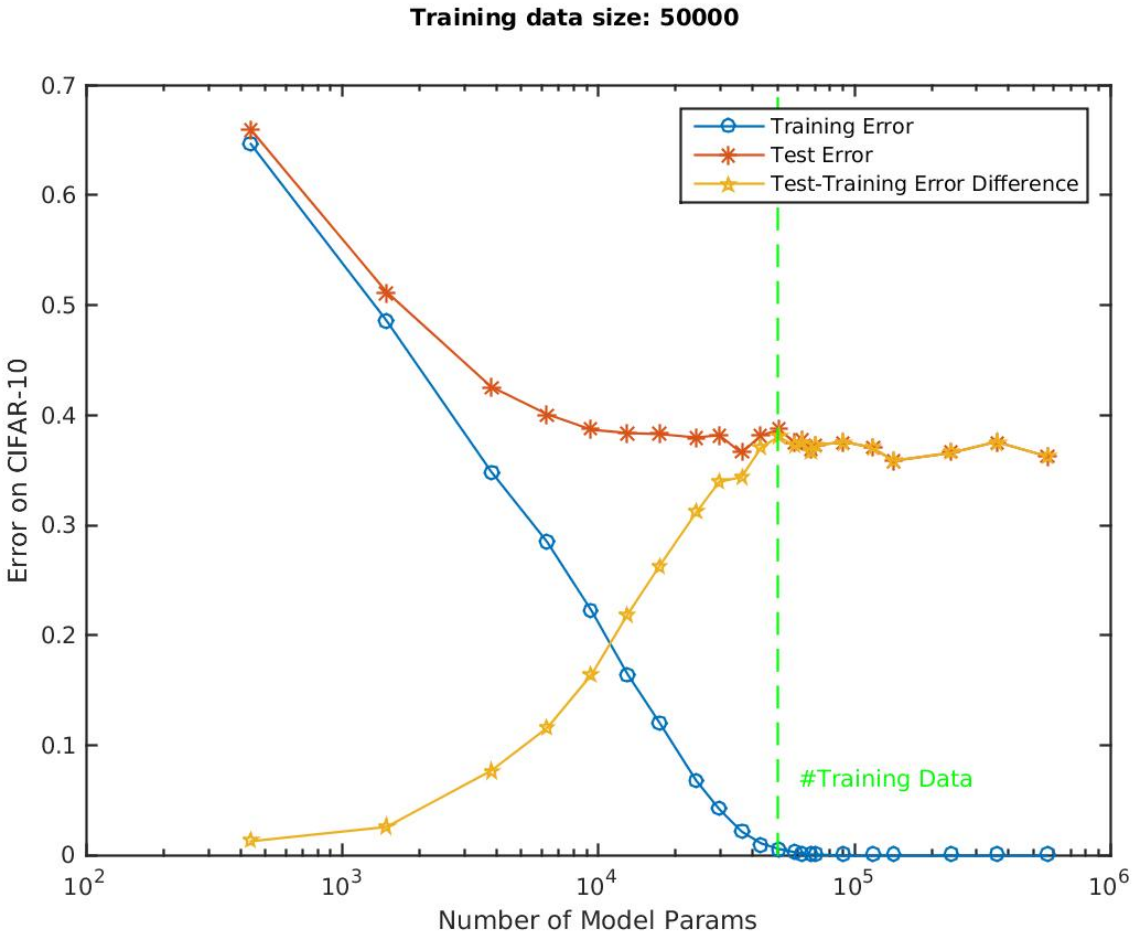


Figure 3: The previous figures show dependence on N – number of training examples – for a fixed architecture with W parameters. This figure shows dependence on W for a fixed training set with N examples. The network is again a 5-layer all convolutional network (i.e., no pooling). All hidden layers have the same number of channels. Neither data augmentation nor regularization is performed. The classical theory explains the generalization behavior on the left; the challenge is to explain the lack of overfitting for $W > N$.As expected from Theory II[2], there is zero error as soon as $W = N$ and for $W \geq N$.

corresponding to the unique zero that exists for for $n = W - 1$. Among them SGD chooses the one with the largest volume and thus with the largest margin.

Notice that overparametrization *does not necessarily improve the test error* which is optimal around the the boundary between over- and under-parametrization.

We consider the steps of our argument in turn, *starting with properties of SGD that have been so far unrecognized* from the machine learning point of view, to the best of our knowledge.

5.1 SGD as an approximate Langevin equation

We consider the usual SGD update defined by the recursion

$$f_{t+1} = f_t - \gamma_t \nabla V(f_t, z_t), \tag{6}$$

where z_t is fixed, $\nabla V(f_t, z_t)$ is the gradient of the loss with respect to f at z_t, and γ_t is a suitable decreasing sequence. When $z_t \subset [n]$ is a minibatch, we overload the notation and write $\nabla V(f_t, z_t) = \frac{1}{|z_t|} \sum_{z \in z_t} \nabla V(f_t, z)$.

We define a noise "equivalent quantity"

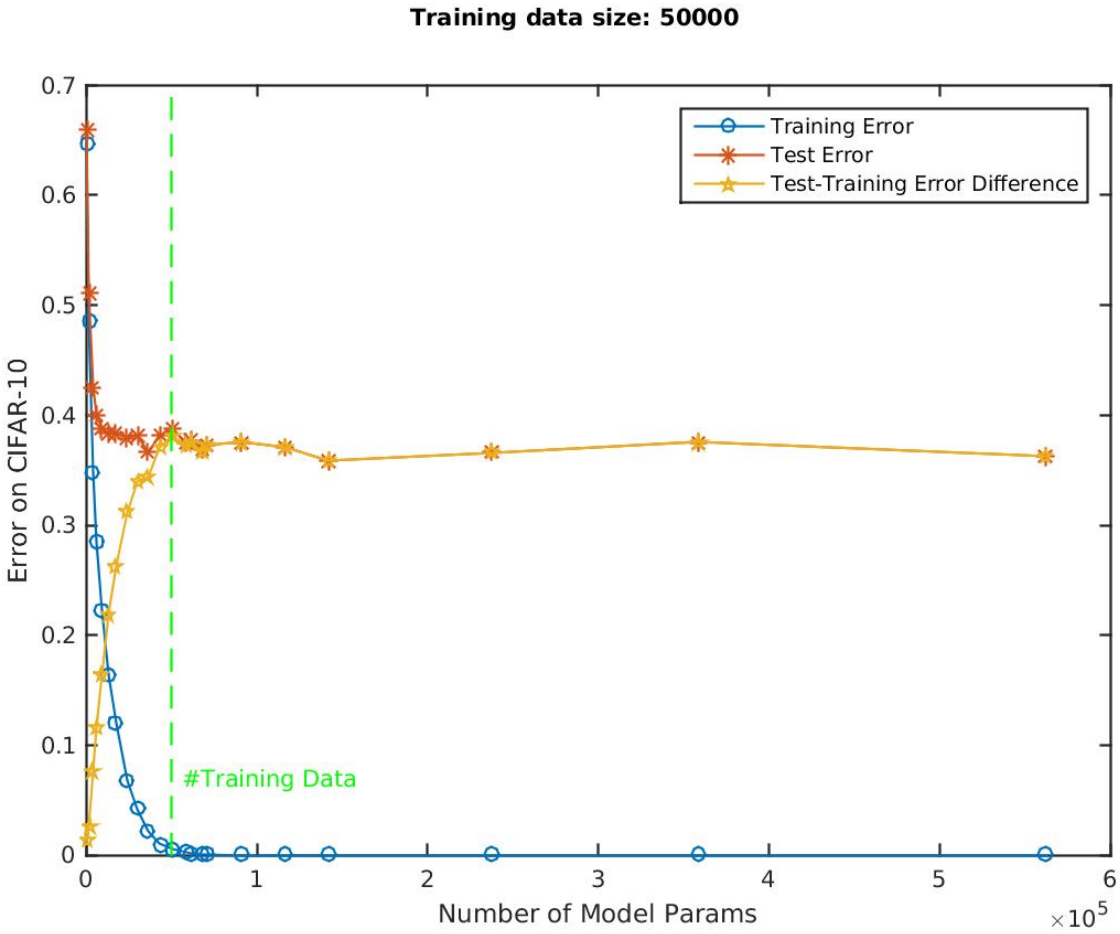


Figure 4: The plot is the same as Figure 3 but with a linear scale for the number of parameters W.

$$\xi_t = \nabla V(f_t, z_t) - \nabla I_{S_n}(f_t), \tag{7}$$

and it is clear that $\mathbb{E}\xi_t = 0$.

We write Equation 6 as

$$f_{t+1} = f_t - \gamma_t(\nabla I_{S_n}(f_t) + \xi_t). \tag{8}$$

With typical values used in minibatch (each minibatch corresponding to z_t) training of deep nets, it turns out that the vector of gradient updates $\nabla V(f_t, z_t)$ empirically shows components with approximate Gaussian distributions (see Figure 5). This is expected because of the Central Limit Theorem (each minibatch involves sum over many random choices of datapoints).

Now we observe that (8) is a discretized Langevin diffusion, albeit with a noise scaled as γ_n rather than $\sqrt{\gamma_n}$. In fact, the continuous SGD dynamics corresponds to a stochastic gradient equation using a potential function defined by $U = I_{S_n}[f] = \frac{1}{n}\sum_{i=1}^{n} V(f, z_i)$ (see Proposition 3 and section 5 in [16]). If the noise is the derivative of the Brownian motion, it is a Langevin equation – that is a stochastic dynamical system – with an associated Fokker-Planck equation on the probability distributions of f_t. The asympotic probability distribution is the Boltzman distribution that is $\approx e^{\frac{-U}{\gamma K}}$.

For more details, see for instance section 5 of [17]. Several proofs that adding a white noise term to equation (6) will make it converge to a global minimum are available (see [18]). Notice that the discrete version of the Langevin dynamics is equivalent to a Metropolis-Hastings algorithm for small learning rate (when the rejection step can be neglected).

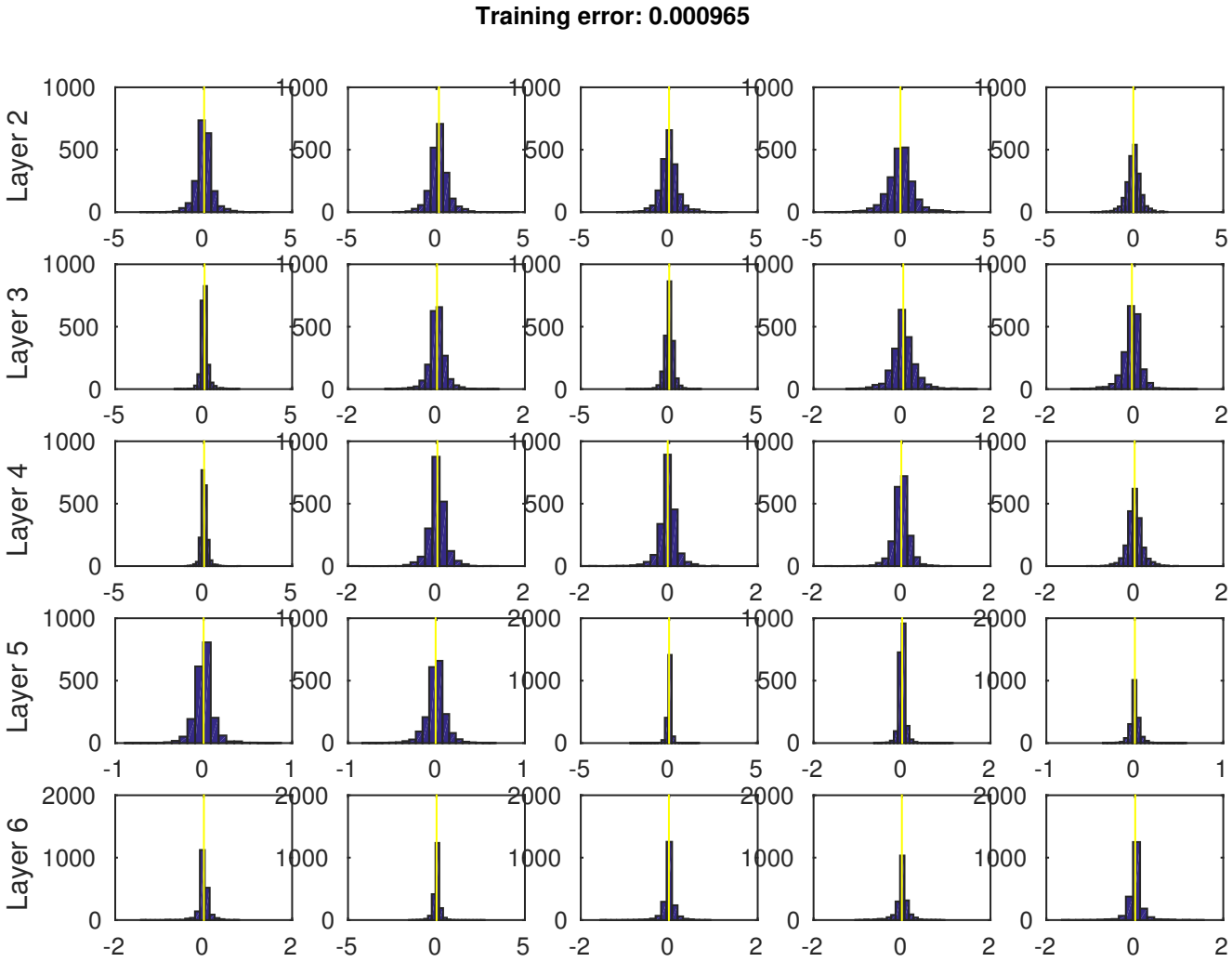


Figure 5: Histograms of some of the components of $\nabla V(f_t, z_i)$ over i for fixed t in the asymptotic regime. Notice that the average corresponds to the gradient of the full loss and that is empirically very small. The histograms look approximatively Gaussian as expected (see text) for minibatches that are not too small or too large.

5.2 SGDL concentrates at large volume, "flat" minima

The argument about convergence of SGDL to large volume minima tha we call "flat", is straighforward. The asymptotic distribution reached by a Langevin equation (GDL) –as well as by SGDL – is the Boltzman distribution that is

$$p(f) = \frac{1}{Z} e^{-\frac{U}{T}}, \tag{9}$$

where Z is a normalization constant, U is the loss and T reflects the noise power. The equation implies, and Figure 8 shows, that SGD prefers degenerate minima relative to non-degenerate ones of the same depth. In addition, among two minimum basins of equal depth, the one with a larger volume, is much more likely in high dimensions (Figure 7). Taken together, these two facts suggest that SGD selects degenerate minimizers and, among those, the ones corresponding to larger isotropic flat regions of the loss. Suppose the landscape of the empirical minima is well-behaved in the sense that deeper minima have broader basin of attraction.Then it is possible to prove that SDGL shows concentration – *because of the high dimensionality* – of its asymptotic distribution Equation 9 – to minima that are the most robust to perturbations of the weights. Notice that these assumptions are satisfied in the zero error case: among zero-minimizer, SGDL selects the ones that are flatter, i.e. have the larger volume[2].

Conjecture 2. *Under regularity assumptions on the landscape of the empirical minima, SGDL corresponds to the following robust*

[2]Given a zero minimizer there is no other minimizer that has smaller volume AND is deeper.

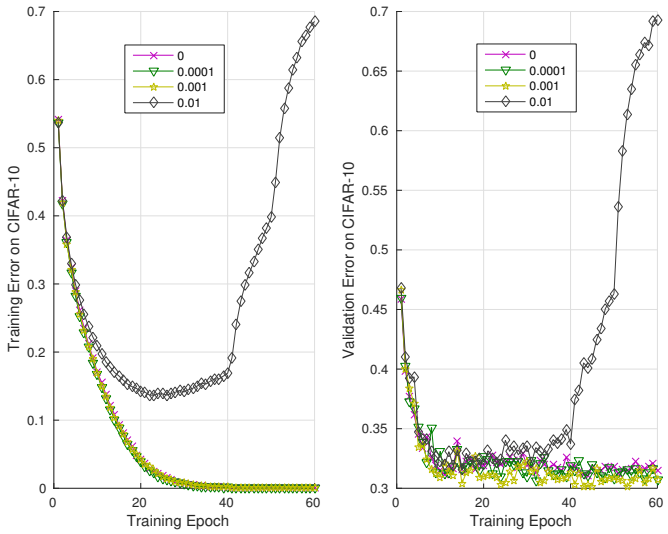


Figure 6: Equation 5 – that is SGD with added Gaussian (with constant power) – behaves in a similar way to standard SGD. Notice that SGDL has slightly better validation performance than SGD.

optimization

$$\min_{w} \max_{(\delta_1,\cdots,\delta_n)} \frac{1}{n} \sum_{i=1}^{n} V(y_i, f_{w+\delta_i}(\mathbf{x_i})). \tag{10}$$

Notice that isotropic and non-isotropic degeneracy is expected to be the key factor for SGDL to converge to zero-minimizers. It is especially important to note that *SGDL and SGD maximize volume and "flatness" of the loss in weight space*. Given a flat minimum, one may ask where SGD will converge to. For situations such as in Figure 9 and for a minimum such as in Figure 10, arguments similar to Appendix 9.6 may hold approximately, suggesting a locally minimum norm solution (see Equation 31). In particular, the weight values found by SGD are expected to be mostly around their average value over the flat region (at least in the case of square loss). We can also consider the polynomial describing $f(x)$ – the function computed by the network when each RELU is replaced by a univariate polynomial approximant. In this case $f(x)$ is a linear function in the vector X comprising all the relevant monomials, implying that SGD converges to the minimum norm solution in X.

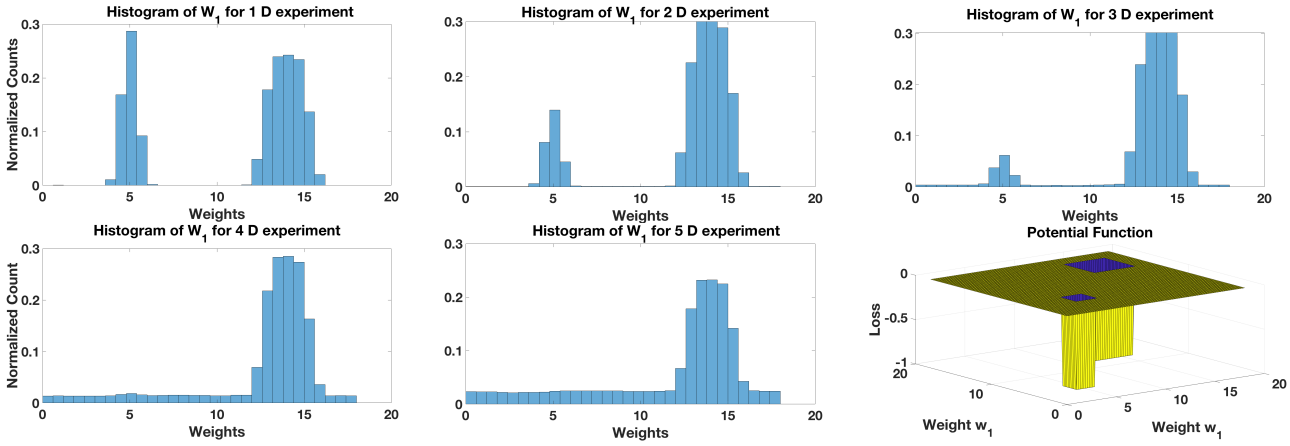


Figure 7: The figure shows the histogram of a one-dimensional slice of the asymptotic distribution obtained by running Langevin Gradient Descent (GDL) on the potential surface on the right. The potential function has two minima: they have the same depth but one has a flat region which is a factor 2 larger in each of the dimensions. The 1D histogram for the first weight coordinate is shown here for dimensionality $1, 2, 3, 4$ and 5D. The figures graphically show – as expected from the asymptotic Boltzman distribution – that noisy gradient descent selects with high probability minimizers with larger margin. As expected, higher dimensionality implies higher probability of selecting the flatter minimum.

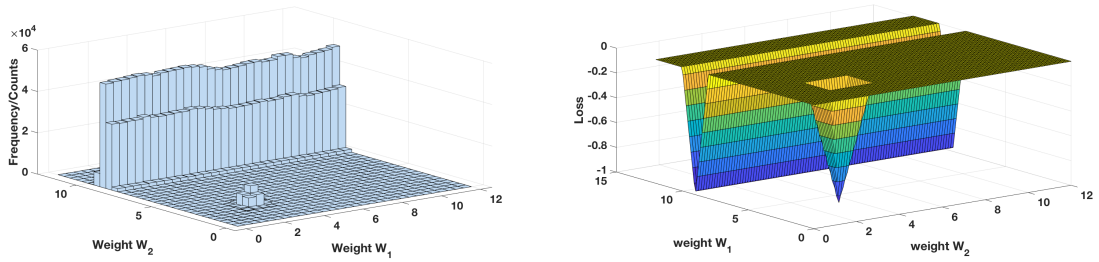


Figure 8: Langevin Gradient Descent (GDL) on the 2D potential function shown above leads to an asymptotic distribution with the histogram shown on the left. As expected from the form of the Boltzman distribution, the Langevin dynamics prefers degenerate minima to non-degenrate minima of the same depth. In high dimensions we expect the asymptotic distribution to concentrate strongly around the degenerate minima as confirmed on figure 9.

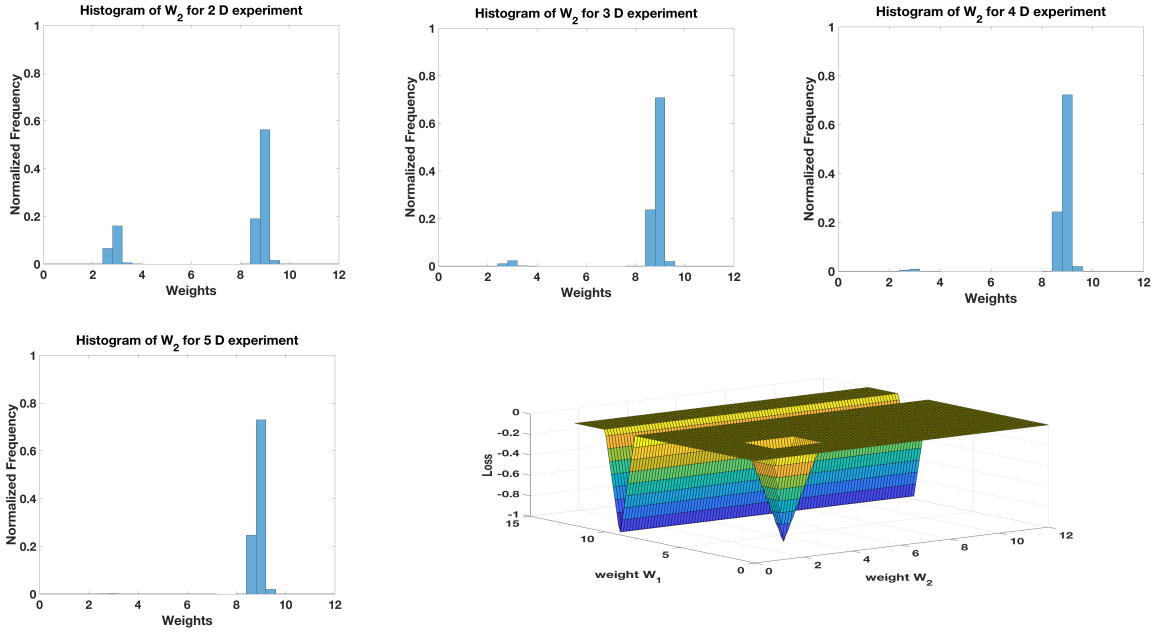


Figure 9: The figure shows the histogram of a one-dimensional slice of the asymptotic distribution obtained by running Langevin Gradient Descent (GDL) on the potential surface on the right. As expected from the form of the Boltzman distribution, the Langevin dynamics prefers degenerate minima to non-degenrate minima of the same depth. Furthermore, as dimensions increase the distribution concentrates strongly around the degenerate minima. This can be appreciated from the figure because the histogram density at $W_1 = 2$ (the degenerate minimum) decreases in density rapidly as dimensions increases.

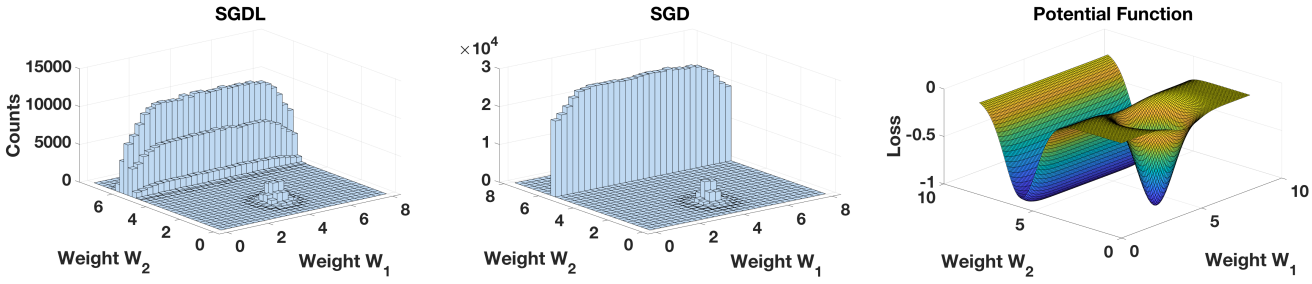


Figure 10: Stochastic Gradient Descent and Langevin Stochastic Gradient Descent (SGDL) on the 2D potential function shown above leads to an asymptotic distribution with the histograms shown on the left. As expected from the form of the Boltzman distribution, both dynamics prefers degenerate minima to non-degenerate minima of the same depth.

5.3 SGD maximizes geometrical margin

Traditionally the functional margin of a correctedly classified data point y_i, x_i is defined as the quantity $\gamma = y_i f(x_i)$. For a linear classifier $f(x) = Wx$, the geometrical margin is the distance *in input space* from the classification boundary and is maximized by minimizing $||W||_2$.

In the nonlinear case it is difficult to relate the functional margin $y_i f(x_i)$ to the geometrical margin. We can use the "linear approximation" provided by Equation 31 which is a local approximation depending on the specific x. Consider an approximation of the geometrical margin of the point x_i as $f(x) = \Lambda_{w,x}\hat{\mathbf{w}}\hat{\mathbf{x}} = W_{x_i}x_i$ which is also a definition of W_{x_i} in terms of $\hat{\mathbf{w}}$ and $\hat{\mathbf{x}}$. where Λ is a diagonal matrix with $0, 1$ components indicating which paths are switched on for x_i. Within the validity of the approximation, larger flatness of W implies greater geometrical margin.

A perhaps more satisfyng approach is the following: propose a new definition of margin that directly capture the idea of measuring robustness to perturbations, show that it is equivalent to the old definition of geometrical margin in the linear case. The new definition we propose is

Definition 1. *For a given functional margin $y_i f(x_i)$, the geometrical margin is*

$$\gamma = max_{||\Delta x||_2} \mid sign(y_i f(x_i)) = sign(y_i f(x_i + \Delta x)). \tag{11}$$

An approximate relation between flatness in the weights and geometrical margin can be obtained by considering an open path j for the component x_i as

$p_j(x_i) = w_1 \cdots w_k x_i$.

Consider the effect of perturbing the weights in the case $k = 3$

$\Delta p_j(x_i) = w_2 w_3 \delta w_1 + w_1 w_3 \delta w_2 + w_1 w_2 \delta w_3$. Assume $\Delta p_j(x_i) \leq \epsilon^2$ for perturbations $\delta w_i \leq 1$. This allows us to bound each of the weights, that is $w_i \leq \epsilon$. This in turn bounds the path weight and the change in p_j induced by a Δx_i perturbation as $\Delta p_j(x_i) = w_1 \cdots w_k \Delta x_i \leq \epsilon^k \Delta x_i$. This implies in this linear approximation that the margin $\gamma > \Delta p_j(x_i)\epsilon^{-k}$ is large if the minimum of the loss is flat wrt the weights.

Another approach to show that SGD provides large margin is is based on the equivalence between SGD and forms of robust optimization and is presented in Appendix 9.3. The proofs are rigorous only in the case of linear networks and approximate otherwise.

Thus SGD selects minimizers corresponding to maximum geometrical margin. Notice that within a single flat minimum the average of the asymptotic fluctuations for each of the degenerate directions (and the non-degenerate ones) is at the maximum margin (the variance however is expected to increase with time).

5.4 "Equivalence" between perturbations in weights and in input

In this section we consider the following fully-connected model for our neural networks: suppose that $D, H, L \in \mathbb{N}$, $x \in \mathbb{R}^D$, $W_1 \in \mathbb{R}^{H \times D}$, $W_2, \ldots, W_{L-1} \in \mathbb{R}^{H \times H}$, $W_L \in \mathbb{R}^{1 \times H}$, $W = (W_1, \ldots, W_L$, and let

$$f_W(x) = \sigma(W_L \sigma(W_{L-1} \cdots W_1 \sigma(x))),$$

where σ denotes the unit-wise ReLU activation. In each of layers $1, 2, \ldots, L-1$, there are H unites, and for $1 \leq j \leq L-1$, we denote the H units by $v_{j,1}, \ldots, v_{j,H}$. We let the output unit (which computes $f_W(x)$) be denoted by v_L, and the D input units be $v_{0,1}, \ldots, v_{0,D}$.

Given a path $v_{0,i_0} \to v_{1,i_1} \to v_{2,i_2} \to \cdots \to v_{L-1,i_{L-1}} \to v_L$, we can also express f_W as a sum over all paths, as follows:

$$f_W(x) = \sum_{i_0 \in [D], i_1, \ldots, i_{L-1} \in [H]} x_{i_0} \prod_{k=0}^{L-1} w(v_{k,i_k} \to v_{k+1,i_{k+1}}) \cdot g_{W,x}(v_{0,i_0}, \ldots, v_L),$$

where $g_{W,x}(v_{0,i_0}, \ldots, v_L) \in \{0, 1\}$ denotes whether or not the path defined by $v_{0,i_0}, \ldots, v_L$ is active for x and W. Note that $f_W(x)$ is a continuous function of both W, x, and is differentiable with respect to x, W almost everywhere on $\mathbb{R}^D \times \mathbb{R}^{H \times D} \times (\mathbb{R}^{H \times H})^{L-1} \times \mathbb{R}^{1 \times H}$ (with respect to the Lebesgue measure). Therefore, for almost all x, W, we can write, for $1 \leq i_0 \leq D$,

$$\frac{\partial f_W(x)}{\partial x_{i_0}} = \sum_{i_1, \ldots, i_{L-1} \in [H]} \prod_{k=0}^{L-1} w(v_{k,i_k} \to v_{k+1,i_{k+1}}) \cdot g_{W,x}(v_{0,i_0}, \ldots, v_L),$$

and for $0 \leq k \leq L-1$, $i_k, i_{k+1} \in [H]$, (*except for the corner cases $k = 0, L-1$...*),

$$\frac{\partial f_W(x)}{\partial w(v_{k,i_k} \to v_{k+1,i_{k+1}})} = \sum_{i_0 \in [D], i_1, \ldots, \hat{i}_k, \ldots i_{L-1} \in [H]} x_{i_0} \prod_{k' \in [0,L-1], k' \neq k} w(v_{k',i'_k} \to v_{k'+1,i_{k'+1}}) \cdot g_{W,x}(v_{0,i_0}, \ldots, v_L),$$

where a hat denotes exclusion of the index.

For convenience we write $\mathcal{W} = \mathbb{R}^{H \times D} \times (\mathbb{R}^{H \times H})^{L-1} \times \mathbb{R}^{1 \times H}$

At a high level, our aim is to show that if a small change in W does not significantly change the value of $f_W(x)$ (uniformly, for all x in some bounded set), then for all x in that bounded set, a small change in x does not change the value of $f_W(x)$, for the given W. In order to show this rigorously, we must assume that the values of $g_{W,x}$ do not change for any of these perturbations of W. In addition, we need a bound on the second-order derivative of f. Formally, we assume, that for some W_0, there is a set $\mathcal{X} \subseteq \mathbb{R}^d$ and a set $\mathcal{W}' \subseteq \mathcal{W}$, $W \in \mathcal{W}'$ such that

1. $\|\nabla^2_W f_W(x)\|_\sigma \leq M$ for all $W \in \mathcal{W}', x \in \mathcal{X}$. (Here $\|A\|_\sigma$ denotes the spectral norm of a matrix A.)
2. There exist r, δ such that the following is true. For all $x \in \mathcal{X}$ such that $f_W(x)$ is differentiable with respect to x and W for all W in an open set U_x containing W, there is a ball $B(W_0, r) \subseteq \mathcal{W}'$ (with respect to the Euclidean norm in $\mathcal{W}$), such that for all $W' \in B(W_0, r)$,
$$|f_{W'}(x) - f_{W_0}(x)| \leq \delta.$$
3. For each possible combination of the activations $g_{W,x}(v_{0,i_0}, \ldots, v_L) = A_{i_0,\ldots,i_{L-1}} \in \{0, 1\}$ (over all $i_0, \ldots, i_{L-1}$), there is some $x \in \mathcal{X}$ such that $g_{W,x}(v_{0,i_0}, \ldots, v_L) = A_{i_0,\ldots,i_{L-1}}$ for all $i_0, \ldots, i_{L-1}$. (TODO: this is unrealistic, can probably be relaxed...)

We show that there is some $r' > 0, \delta' > 0$ such that for all $x \in \mathcal{X}$, for all $x' \in B(x, r') \cap U_x$ (TODO: can make U_x bigger here),
$$|f_{W_0}(x) - f_W(x')| \leq \delta'.$$

5.5 Large margin implies good classification performance

This is for now a short section. There are a number of well-known situations where large margin clearly implies good classification performance. The classical examples are Perceptrons and SVMs for which formal results are available. It is more difficult to apply existing results on large margin classifiers to deep networks in order to obtain quantitative and useful bounds. We hope to present explicit bounds in a future update of this memo.

The arguments in this version of the paper connecting margin with expected error are qualitative and explains why there is no overfitting but they do not provide bounds that are quantitatively applicable.

The main argument is provided by the robustness-based bounds of Appendix 8.3 and the theorem there.

6 SGD puzzles

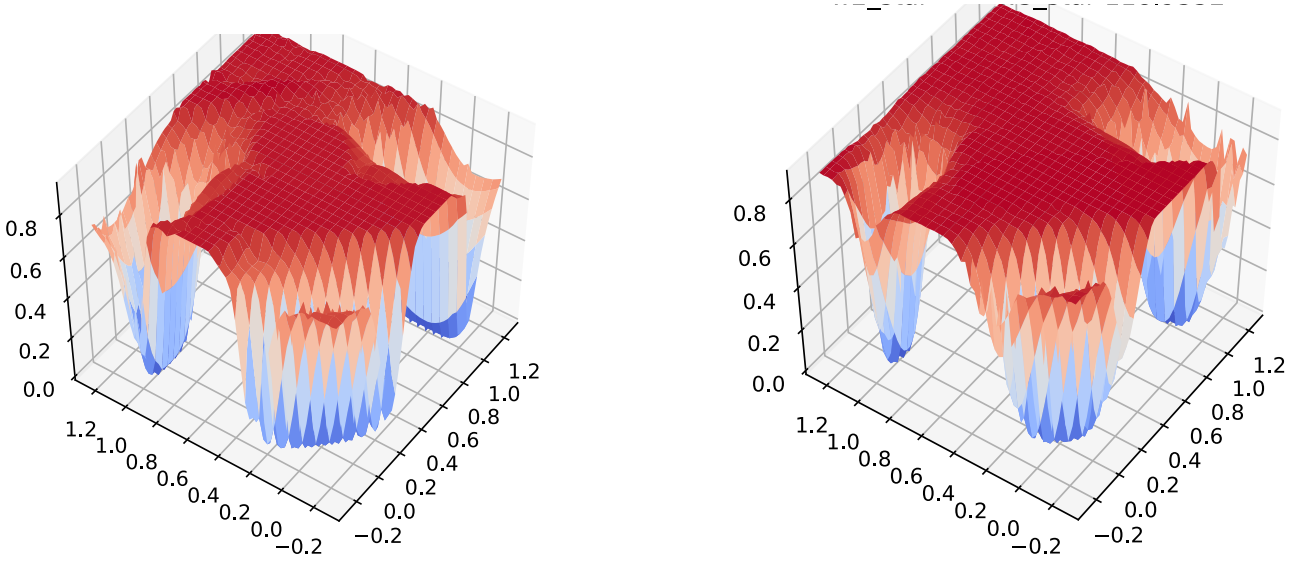
Two puzzling properties of deep networks that we can now account for are

- the no generalization behavior tested with training data with random labels, see[4], while expected error is much better than chance for natural labels;
- the behavior shown in Figure 13 and Figure 15: the expected error, which decreases with increasing training set size, is not affected by increasing the number of parameters beyond the training data size.

6.1 Random Labels

In the first case, Theory II predicts that it is in fact possible to interpolate the data on the training set, that is to achieve zero empirical error (because of overparametrization) and that this is in fact easy – because of the very high number of zeros of the polynomial approximation of the network– assuming that the target function is in the space of functions realized by the network. For n going to infinity we expect that the empirical error will converge to the expected (chance), as shown in Figures 1 and 2. For finite n when $n < W$, the fact that the empirical error (which is zero) is so different from the expected is puzzling, as observed by [4], especially because the algorithm is capable of low expected error with the same n for natural labels (and also when trained with random labels but tested with perturbations of the training images, see Figures in Appendix 9.5).

The puzzle is explained in terms of the distribution-dependent geometrical margin. Our theory suggests that SGD maximizes margin. A larger margin is in fact found for natural labels than for random labels as shown in Table 1 and in Figure 11 and Figure 12. Figure 11 shows "three-point interpolation" plots to illustrate the flatness of the landscape around global minima of the empirical loss found by SGD, on CIFAR-10, with natural labels and random labels, respectively. Specifically, let w_1, w_2, w_3 be three minimizers for the empirical loss found by SGD. For $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ on the simplex Δ_3, let
$$w_\lambda = \lambda_1 w_1 + \lambda_2 w_2 + \lambda_3 w_3 \tag{12}$$
We then evaluate the training accuracy for the model defined by each interpolated weights w_λ and make a surface plot by embedding Δ_3 in the 2D X-Y plane. As we can see, the natural label case depict a larger flatness region around each of the three minima than the random



(a) natural label

(b) random label

Figure 11: Illustration of the landscape of the empirical loss on CIFAR-10.

	MNIST	CIFAR-10
all params	45.4 ± 2.7	17.0 ± 2.4
all params (random label)	6.9 ± 1.0	5.7 ± 1.0
top layer	15.0 ± 1.7	19.5 ± 4.0
top layer (random label)	3.0 ± 0.1	12.1 ± 2.6

Table 1: The "flatness test": at the minimizer, we move the weights around in a random direction, and measure the furthest distance until the objective function is increased by ε (0.05), and then measure the average distance.

label case. Section 7.1 shows a direct relation between the range of flatness and the norm λ of the perturbations. Note that λ is also the regularization parameter (see section 9.4: *larger λ means greater regularization, which implies better stability (β-stability of regularization has $\beta \approx \frac{K}{\lambda n}$) and better generalization bounds*.

The same phenomenon could be observed more clearly on the MNIST dataset, where the images of the same category are already quite similar to each other in the pixel space, making it more difficult to fit when random labels are used. Therefore, the difference in the characteristics of the landscapes is amplified. As shown in Figure 12, big flat regions could be observed in the natural label case, while the landscape for the random label experiment resembles sharp wells.

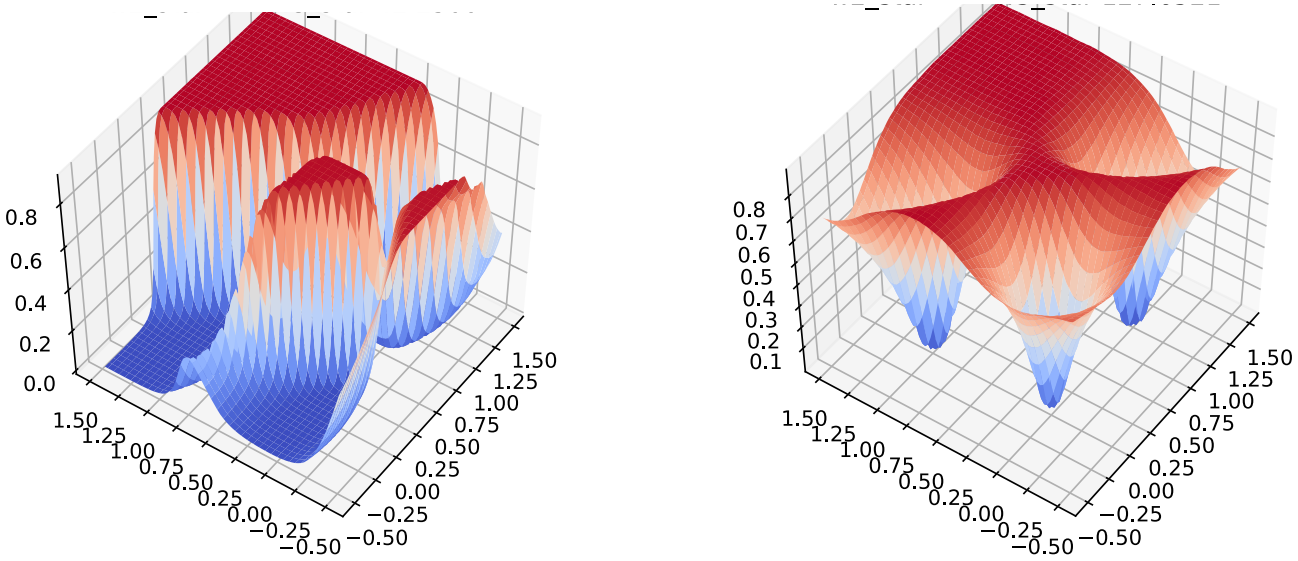
It is difficult to visualize the isotropic flatness of the landscape when the weights are typically in the scale of one million dimensions. To assess the isotropic flatness, we employ the following procedure around a minimum found by SGD: choose a random isotropic direction δw with $\|\delta w\| = 1$, perform a line search to find the "flatness radius" in that direction:

$$r(w, \delta w, \varepsilon) = \sup\{r : |\hat{I}(w) - \hat{I}(w + r\delta w)| \leq \varepsilon\} \tag{13}$$

The procedure is repeated T times and the average radius is calculated. The overall procedure is also repeated multiple times to test the average flatness at different minima. The results are shown in Table 1. For both CIFAR-10 and MNIST, we observe a difference between the natural label and random label.

6.2 No Overfitting with Increasing Network Size

In the second case, what is important for low expectation error is not parameter number but size of the training set. The intuition comes from Theory II: minimization by SGD finds global minima that are similar to minimum norm solutions – which are not changed by increasing the number of equations beyond the number of data points. Local minima instead are determined by the number of parameters since they determine the number of equations for the critical points of the gradient: this implies that increasing the number of parameters much beyond the number of data in the training set, should not affect generalization but may make it easier to find global minima relative to local minima. In summary overparametrization in Figure 4 for $W > n$ does not affect the test error because SGD will disregard degenerate directions.



(a) natural label

(b) random label

Figure 12: Illustration of the landscape of the empirical loss on MNIST.

Architecture	Number of Parameters	Training Accuracy	Test Accuracy
MLP 1x512	1,209,866	100.0	50.51
Alexnet	1,387,786	100.0	76.07
Inception	1,649,402	100.0	85.75
Wide Resnet	8,949,210	100.0	88.21

Table 2: A number of different network architectures are trained on CIFAR-10. We turn off all the regularizers in order to avoid implicitly constraining the hypothesis space size. We found that despite the network size continuously increases, the test performance does not drop, but even improves.

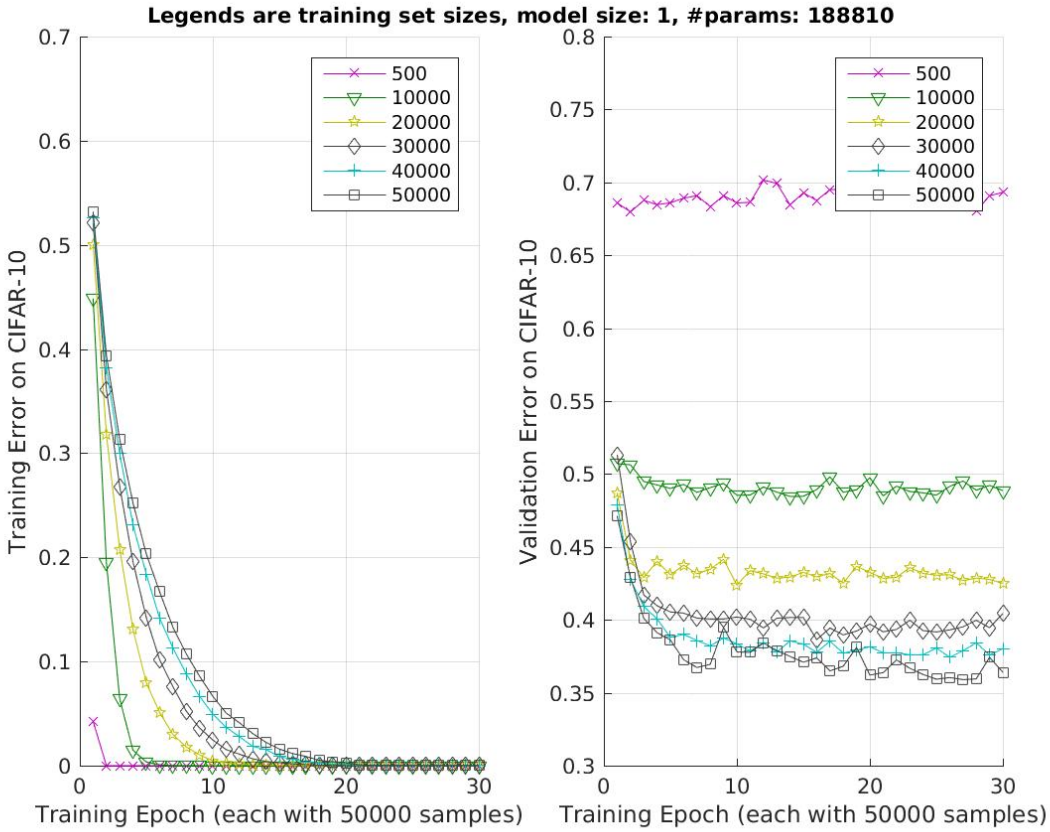


Figure 13: Training with different training set sizes (from 500 to 50,000, one per curve) on CIFAR-10. It shows what happens to empirical and testing error if one severely reduces the number of training examples. There is overfitting and little generalization with small training sets.

7 Discussion

7.1 Comments on Robustness wrt Weights and Robustness wrt Data

A natural intuition, that we discuss further in Appendix 9.7, is that several forms of stability are closely related. In particular, *perturbations of the weights follow from perturbations of the data points in the training set* because the function $f(x)$ resulting from the training is parametrized by the weights that depend on the data points S_n: $f(x) = f(x; w(z_1, \cdots, z_n))$, with $z = (x, y)$[3] . Thus *isotropic flat regions in weight space of the global minimum of the empirical loss indicate stability with respect to the weights; the latter in turn indicates stability with respect to training examples*.

7.2 Summary: why and when SGD has low expected error and why and when it generalizes

We conjecture that low expected error in the regime of $N < W$ and of generalization in the regime $N > W$ exhibited by deep convolutional networks in multi-class tasks such as CIFAR10 and Imagenet are due to three main factors:

- the implicit robustifying eg regularizing effect of SGD
- the task is compositional
- the task is multiclass.

[3] In a differentiable situation, one would write $df = \frac{df}{dw}\frac{dw}{dS}dS$ and $dw = \frac{dw}{dS}dS$. The latter equation would show that perturbations in the weights depend on perturbations of the training set S.

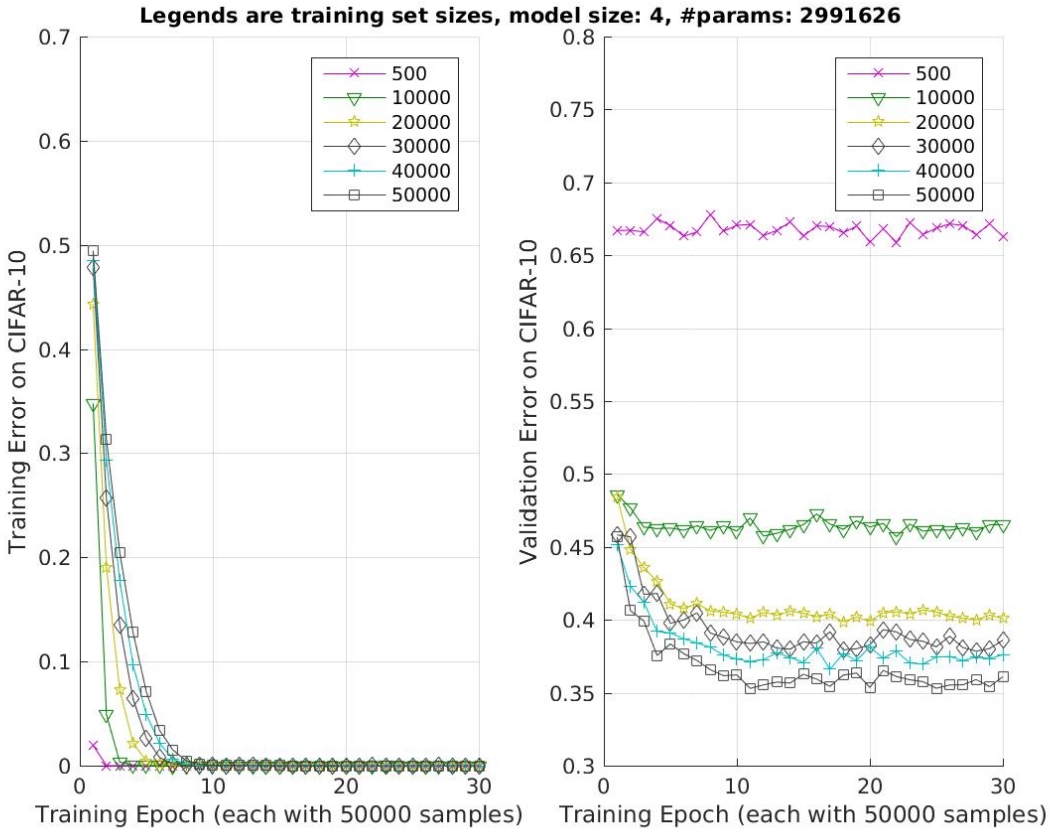


Figure 14: Training with different training set sizes (from 500 to 50,000, one per curve) on CIFAR-10. This is a larger model with more parameters.

Speed of convergence of SGD is not the deep reason for good generalization. We expect the speed of convergence to be correlated with good generalization because convergence will depend on the relative size of the basins of attraction of the minima of the empirical risk, which in turn depend on the ratio between the effective dimensionality of the minima and the ambient dimensionality. Theory III in this paper, together with Theory II, answers the question about the unusual properties of Stochastic Gradient Descent used for training overparametrized deep convolutional networks. SGDL and SGD select with high probability solutions with zero or small empirical error (Theory II) – because they are flatter. It is almost a miracle that these solutions are also, as we have shown, the best in achieving low expected error and generalization – because they have the largest margin, are the most stable wrt parameters, most stable wrt training data and most stable with respect to the x value.

7.3 Open questions

The main task is to quantify the qualitative characterization of generalization and consitency of convolutional deep networks given here. In particular, which non-vacuous bounds may relate margin to expected error? Can we spell conditions on data distributions that yield large margin and low expected error? Under which hypotheses is the best expected error reached for $W = N$? Can we improve SGD or find optimal parameters such as mini-batch size?

Acknowledgment

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF – 1231216. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the DGX-1 used for this research.

References

[1] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep - but not shallow - networks avoid the curse of dimensionality: a review," tech. rep., MIT Center for Brains, Minds and Machines, 2016.

[2] P. T. and Q. Liao, "Theory ii: Landscape of the empirical risk in deep learning," tech. rep., MIT Center for Brains, Minds and Machines, 2017.

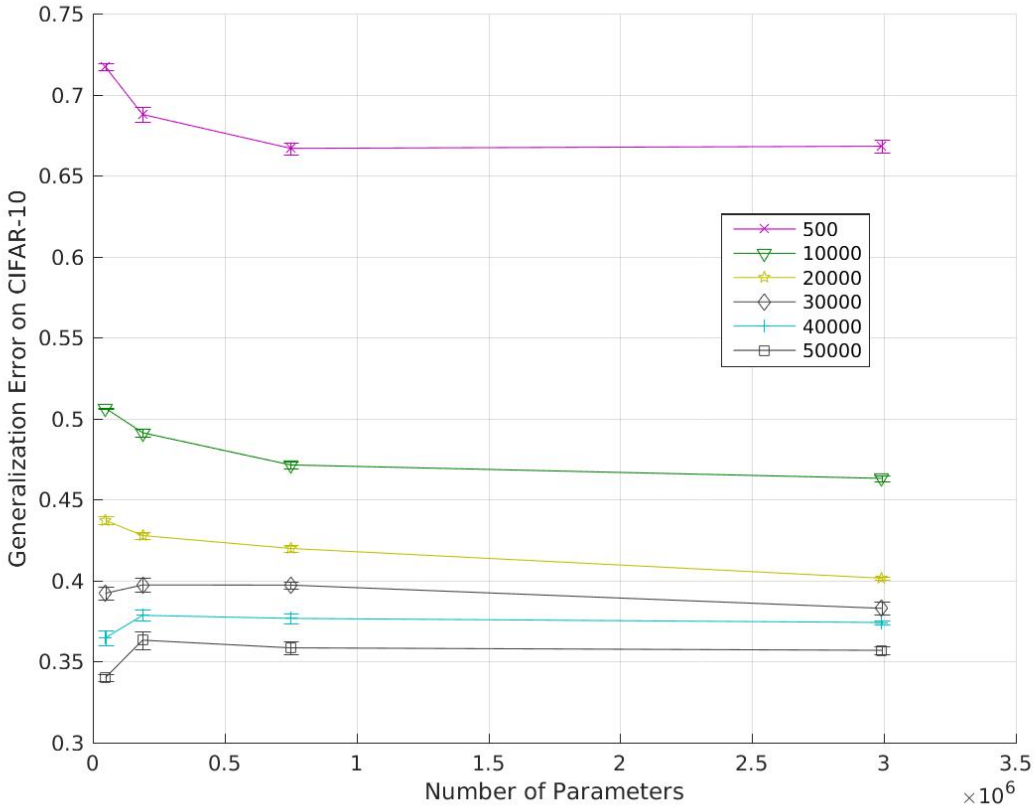


Figure 15: Generalization error as a function of number of parameters in the network. Generalization error is defined as the difference between training error and validation error. We train a 4-hidden-layer convolutional network with different training set sizes on CIFAR-10. Different curves correspond to different training set sizes (from 500 to 50,000). All models are trained 30 epochs. Increasing the number of parameters does not hurt generalization.

[3] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge eBooks, 2014.

[4] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Learning Representations (ICLR)*, 2017.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, pp. 436–444, 2015.

[6] K. Fukushima, "Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[7] D. Hubel and T. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, p. 106, 1962.

[8] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, pp. 1019–1025, Nov. 1999.

[9] 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," *arXiv:1611.01838 [cs]*, Nov. 2016. arXiv: 1611.01838.

[10] 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," *arXiv:1609.04836 [cs, math]*, Sept. 2016. arXiv: 1609.04836.

[11] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, "Sharp minima can generalize for deep nets," *arXiv preprint arXiv:1703.04933*, 2017.

[12] G. Karolina Dziugaite and D. M. Roy, "Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data," *ArXiv e-prints*, Mar. 2017.

[13] S. Ma and M. Belkin, "Diving into the shallows: a computational perspective on large-scale shallow learning," *ArXiv e-prints*, Mar. 2017.

[14] S. Gelfand and S. Mitter, "Recursive stochastic algorithms for global optimization in R^d," *Siam J. Control and Optimization*, vol. 29, pp. 999–1018, September 1991.

[15] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks* (D. Saad, ed.), Cambridge, UK: Cambridge University Press, 1998. revised, oct 2012.

[16] D. Bertsekas and J. Tsitsiklis, "Gradient Convergence in Gradient methods with Errors," *SIAM J. Optim.*, vol. 10, pp. 627–642, Jan. 2000.

[17] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[18] B. Gidas, "Blobal optimization via the Langevin equation," *Proceedings of the 24th IEEE Conference on Decision and Control*, pp. 774–778, 1985.

[19] M. Anthony and P. Bartlett, *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, 2002.

[20] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

[21] H. Xu and S. Mannor, "Robustness and generalization," *CoRR*, vol. abs/1005.2243, 2010.

[22] V. Koltchinskii and D. Panchenko, "Empirical margin distributions and bounding the generalization error of combined classifiers," *Annals of Statistics*, pp. 1–50, 2002.

[23] N. Srebro, K. Sridharan, and A. Tewari, "Smoothness, low noise and fast rates.," in *NIPS* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), pp. 2199–2207, Curran Associates, Inc., 2010.

[24] S. Gelfand and S. Mitter, "Recursive Stochastic Algorithms for Global Optimization in $\mathbb{R}^d $," *SIAM J. Control Optim.*, vol. 29, pp. 999–1018, Sept. 1991.

[25] S. B. Gelfand and S. K. Mitter, "Metropolis-Type Annealing Algorithms for Global Optimization in R^d," *SIAM Journal on Control and Optimization*, vol. 31, no. 1, pp. 111–131, 1993.

[26] S. Rajasekaran, "On the Convergence Time of Simulated Annealing," *Technical Reports (CIS)*, Nov. 1990.

[27] M. Raginsky, A. Rakhlin, and M. Telgarsky, "Non-convex learning via stochastic gradient langevin dynamics: A nonasymptotic analysis," *arXiv:180.3251 [cs, math]*, 2017.

[28] S. Mandt, M. D. Hoffman, and D. M. Blei, "A Variational Analysis of Stochastic Gradient Algorithms," *arXiv:1602.02666 [cs, stat]*, Feb. 2016. arXiv: 1602.02666.

[29] F. Anselmi, L. Rosasco, and T. Tan, C.and Poggio, "Deep Convolutional Networks are Hierarchical Kernel Machines," *Center for Brains, Minds and Machines (CBMM) Memo No. 35, also in arXiv*, 2015.

[30] B. Neyshabur, R. Tomioka, and N. Srebro, "Geometry of Optimization and Implicit Regularization in Deep Learning," *arXiv preprint arXiv:1705.03071*, 2017.

[31] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton Series in Applied Mathematics, Princeton University Press, October 2009.

[32] H. Xu, C. Caramanis, and S. Mannor, "Robustness and regularization of support vector machines," *J. Mach. Learn. Res.*, vol. 10, pp. 1485–1510, Dec. 2009.

[33] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[34] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 463–482, 2002.

[35] S. M. Kakade, K. Sridharan, and A. Tewari, "On the complexity of linear prediction: Risk bounds, margin bounds, and regularization," in *Advances in neural information processing systems*, pp. 793–800, 2009.

[36] B. M. Lake, R. Salakhutdinov, and J. B. Tenenabum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[37] A. Maurer, "Bounds for Linear Multi-Task Learning," *Journal of Machine Learning Research*, 2015.

[38] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *CoRR*, vol. abs/1611.03530, 2016.

[39] H. N. Mhaskar, "Approximation properties of a multilayered feedforward artificial neural network," *Advances in Computational Mathematics*, vol. 1, pp. 61–80, 1993.

[40] T. Poggio, "On optimal nonlinear associative recall," *Biological Cybernetics*, vol. 19, pp. 201–209, 1975.

[41] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, pp. 53–58, 1989.

[42] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *CoRR*, vol. abs/1312.6120, 2013.

[43] K. Kawaguchi, "Deep learning without poor local minima," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.

8 First Appendix: classical generalization bounds for $N > W$

8.1 Classical generalization bounds

The simplest and oldest bound is provided by theorem 16.2 in [19] which provides the following sample bound for a generalization error ϵ_G with probability at least $1 - \delta$ in a network in which the W parameters (weights and biases) that minimize the empirical error (the theorem is stated in the standard ERM setup) are expressed in terms of k bits:

$$M(\epsilon_G, \delta) \leq \frac{2}{\epsilon_G^2} \left(kW \log 2 + \log \left(\frac{2}{\delta} \right) \right) \tag{14}$$

An interesting comparison between shallow and deep compositional networks based on this bound is described in Appendix 9.1.

8.2 Covering number of the space explored by SGD

Definition 2 (Covering Number)**.** *The covering number $N_\varepsilon(K)$ for a set K is the minimum cardinality of a set C, such that*

$$K \subset \bigcup_{x \in C} B(x; \varepsilon)$$

where $B(x;\varepsilon)$ denotes a metric ball of radius ε centered at x. The metric should be clear from the context unless otherwise specified.

The motivation is that when we run an actual learning algorithm (e.g. SGD) on a hypothesis space $\mathcal{H}$, the algorithm that runs in finite time might only be able to explore a subset K. Therefore the effective hypothesis space is K instead of $\mathcal{H}$. We would like to characterize $N_\varepsilon(K)$, which might be much smaller than $N_\varepsilon(\mathcal{H})$.

Consider the SGD algorithm as defined in (4). If there is an explicit projection step onto the set K, then we could compute $N_\varepsilon(K)$ directly. If there is no explicit projection, then for an algorithm that runs for T steps:

$$\begin{aligned} f_T &= f_{T-1} - \gamma_{T-1} \nabla V(f_{T-1}, z_{T-1}) \\ &= f_0 - \sum_{t=0}^{T-1} \gamma_t \nabla V(f_t, z_t) \end{aligned}$$

The effectively explored space can be controlled if we have a bound on the size of the accumulated gradients,

$$\left\| \sum_{t=0}^{T-1} \gamma_t \nabla V(f_t, z_t) \right\| \leq G_T \tag{15}$$

where G_T should be a universal bound that is independent of the actual sampled training data, and of the randomness of the algorithm. Our formulation should apply to both the case of pure SGDs and multi-epoch SGDs. In this case, it is clear that

$$f_T \in K = B(f_0; G_T)$$

Lemma 1. *In a d dimensional Euclidean space, for $\varepsilon \in (0, r)$, the covering number of a Euclidean ball is bounded as*

$$N_\varepsilon(B(\cdot; r)) \leq \left(\frac{3r}{\varepsilon} \right)^d \tag{16}$$

Remarks Applying the lemma directly with the bound G_T on SGD movements, we get

$$N_\varepsilon(K) \leq \left(\frac{3G_T}{\varepsilon} \right)^d$$

The uniform bounds on generalization are roughly of order

$$O\left(\sqrt{\frac{\log N_\varepsilon(K)}{n}} \right) \leq O\left(\sqrt{\frac{d \log(3G_T/\varepsilon)}{n}} \right)$$

As long as the bound G_T grows slower than $\exp(n)$, then generalization should be possible. Here $T > n$ in the multiple epoch case.

Convex learning with Lipschitz Loss Consider the case of convex learning problem with an L-Lipschitz loss function. With a constant step size $\gamma = R/(L\sqrt{T})$, where T is the total number of (full) gradient steps, and $\|f_0 - f^\star\| \leq R$, then an optimization convergence rate on the loss of $O\left(\frac{RL}{\sqrt{T}}\right)$ could be obtained [20]. In this case,

$$\begin{aligned} \left\| \sum_{t=0}^{T-1} \gamma \nabla V(f_t, z_t) \right\| &\leq \gamma \sum_{t=0}^{T-1} \|\nabla V(f_t, z_t)\| \\ &\leq \frac{R}{L\sqrt{T}} \sum_{t=0}^{T-1} L \\ &\leq R\sqrt{T} \end{aligned}$$

By plugging $G_T \leq R\sqrt{T}$, we get the following generalization bound

$$O\left(\sqrt{\frac{d \log(T/\varepsilon)}{n}} \right)$$

As a result, as long as the number of optimizing steps grows less than exponentially in the number of training samples, generalization could be guaranteed.

Convex learning with smooth loss For the Lipschitz loss, the scale of gradients does not necessarily decay even when approaching the optimal solution (e.g. the absolute value function). For smooth loss, the gradients becomes smaller when we get closer to the optimal. Therefore, the bound on the movements by the gradient methods could be improved.

Furthermore, when we have an explicitly (and fast enough) decay rate of the size of the gradients, we might be able to get finer-grain control by realizing that after some fixed number of steps, the gradients are so small that they will remain within an ε-ball.

8.3 Robustness-based Bounds for Generalization

A way to theoretically connect "flatness" in weight space to existing generalization results, is to invoke the notion of *weak robustness* and to show that it is equivalent to flatness of a minimizer. Theorem 18 in [21] proves that *weak robustness* is necessary and sufficient for generalization. For simplicity, we focus on robustness (instead of weak robustness) since robustness implies generalization which is our focus here. We use the notation of [21].

Definition 3. *Algorithm A is $(K, \epsilon(\cdot))$-robust, for $K \in \mathcal{N}$ and $\epsilon(\cdot) : \mathcal{Z}^n \mapsto \mathcal{R}$, if $\mathcal{Z}$ can be partitioned into K disjoint sets, denoted by $\mathcal{C} = (C_i)_{i=1}^K$, such that the following holds: for any dataset $S \in \mathcal{Z}^n$, $\forall s \in S$ and $\forall z \in \mathcal{Z}, \forall i = 1, \cdots, K$: if $s, z \in C_i$, then*

$$\|V(\mathcal{A}_S, s) - V(\mathcal{A}_S, z)\| \leq \epsilon(S) \tag{17}$$

Theorem 1. *If a learning algorithm A is $(K, \epsilon(\cdot))$-robust, and the training sample set S is generated by N IID draws from μ, then for any $\delta > 0$, with probability at least $1 - \delta$ we have,*

$$|\mathcal{L}(\mathcal{A}_s) - \ell_{lemp}(\mathcal{A}_S)| \leq \epsilon(S) + M\sqrt{\frac{2K \log 2 + 2\log(\frac{1}{\delta})}{n}}, \tag{18}$$

with M being an upper bound on the loss function V.

The key step in the argument about flatness, stability and generalization is to prove that flatness implies robustness as defined above.

We now extend theorem 1 to a data dependent version using a standard union bound. Suppose algorithm A is robust for a countable collection $\mathcal{K}$ of pairs $(K_{\mathcal{C}}, \epsilon_{\mathcal{C}}())$, indexed by $\mathcal{C}$, where $\mathcal{C}$ is the partition of size $K_{\mathcal{C}}$ in the definition of robustness. We have in mind a situation where a coarse partition can be set (in a data-independent way) with a potentially larger function $\epsilon_{\mathcal{C}}$ but smaller $K_{\mathcal{C}}$, while a more fine partition would yield a smaller value of $\epsilon_{\mathcal{C}}$ but a larger $K_{\mathcal{C}}$. Since $\epsilon_{\mathcal{C}}(S)$ is a data-dependent quantity, the optimal choice of the partition can only be performed after seeing the data.

Theorem 2. *If a learning algorithm A is $(K_{\mathcal{C}}, \epsilon_{\mathcal{C}}(\cdot))$-robust for a collection $\mathcal{K}$ of pairs,*

$$\mathbb{P}\left\{|\mathcal{L}(\mathcal{A}_s) - \ell_{lemp}(\mathcal{A}_s)| \leq \inf_{(K_{\mathcal{C}}, \epsilon_{\mathcal{C}}(\cdot)) \in \mathcal{K}} \left[\epsilon_{\mathcal{C}}(S) + M\sqrt{\frac{2K_{\mathcal{C}} \log 2 + 2\log((\pi(\mathcal{C})\delta)^{-1}}{n}}\right]\right\} \geq 1 - \delta \tag{19}$$

for any prior distribution $\pi(\mathcal{C})$ on the collection $\mathcal{K}$.

As an example, consider a collection of axis-aligned grid partitions with granularity γ. If the input space is $[0, 1]^d$, the size of the partition is $(1/\gamma)^d$, and the corresponding function $\epsilon_{\mathcal{C}}$, of course, depends on the algorithm. One can then assign a prior probability proportional to $1/k^2$ for discretization at level $\gamma = 2^{-k}$, in which case the penalty $\pi(\mathcal{C})$ is $\log k$ (as, for instance, in [22]).

In our CIFAR case the key question is how large K is and whether it is smaller than N.

Remarks

The authors in [21] assume that f is a L-layer neural network of the binary tree type with P ReLU units per node and $D = 2^L$ inputs, trained on $\mathcal{Z}_n$ by SGD algorithm $\mathcal{A}_{S_n}$ with loss $V(\mathcal{A}_S, z) = |y - \mathcal{A}_S(x)| = |y - f_S(x)|$, with $z = (y, x)$ and weights taking values on the torus. They define the network as follows. The input to the first layer is

$$x^0 := x; \tag{20}$$

the output of layer $v - 1$ is x^v with components

$$\forall v = 1, \cdots, L - 1 \colon x_i^v := [\sum_{j=1}^{2P} w_{i,,j}^{v-1} x_j^{v-1}]_+; i = 1, \cdots, d_v; \tag{21}$$

where $d^v = P \times 2^{L-v}$

$$f_S(x) = [\sum_{j=1}^{d_{L-1}} w_j^{L-1} x_j^{L-1}]_+. \tag{22}$$

The ReLU nonlinearity is Lipschitz continuous since $[a]_+ - [b]_+ \leq \beta|a-b|$ with $\beta = 1$. *If the weights are bounded* then it follows that

$\sum_{j=1}^{d_v} w_{i,j}^v \leq \alpha \forall v, i$ (thus the largest α is $\alpha \leq 2P\pi$). As an aside, note that, in the case of compositional networks, the use of the path norm should give better bounds than [21]. Lemma 22 in [21] proves then that under these conditions such a multilayer network is robust in the infinity norm and therefore generalizes. As noted by [21] the total number of hidden units does not play a role in the bound. Notice also that the local connectivity of deep local networks (such as convolutional network and more in general networks matched to compositional functions) avoids the appearance of the dimensionality of the inputs x_i.

8.4 Distribution Dependent Generalization because of functional margin

In repeat SGD there is a training set and an empirical risk that we write in the form

$$I_{S_n}(f) = \frac{1}{n} \sum_{i}^{n} V(f, z_i) \tag{23}$$

It is clear from the previous sections that repeat SGD provably generalizes under relatively weak assumptions – similar to the usual conditions of convergence of SGD. The problem is of course to provide bounds on the expected error that are non-vacuous for the regime of $W > N$. Several of the classical bounds are meaningless in this case (see section 8.1), but other proofs are possible (see 8.5).

The observations relevant here are:

- We assume that the space of functions expored by SGD is compact. One way is to assume Lipschitz and convexity. Another is to set as an hypothesis "If f_t remains bounded then...". In practice, boundness of some kind must be assumed. It is in principle guaranteed by numerical limits of computer simulations. A simple way to ensure compactness – as a theoretical trick which will not make any difference in the usual simulations – is to assume that the coordinates are periodic – that is that the domain is a torus. Then standard covering numbers arguments guarantee generalization for $N > W$.
- The following arguments are needed to ensure bounds that are independent of the explicity ratio $\frac{W}{N}$. We remark that SGDL (see later for the theoretical arguments) should asymptotically select among the zero-minimizers the ones which have a large margin. We provide empirical support for large empirical margin in the case of CIFAR in the Figures 17 . We introduce the zero-one loss and the γ-margin empirical zero-one loss which is defined as $I_{S,\gamma}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{y_i h(x_i) < \gamma}$. We then use Theorem 5 in [23]

  Theorem 3. *For any hypothesis class (F), with $|f| \leq b$, and any $\delta > 0$, with probability at least $1 - \delta$, simultaneously for all margins $\gamma > 0$ and all $f \in \mathcal{F}$*

$$I(f) \leq 1.01 I_{S,\gamma}(f) + K \left(\frac{2 \log^3 n}{\gamma^2} \mathcal{R}_n^2 + \frac{2 \log \left(\frac{\log\left(\frac{4b}{\gamma}\right)}{\delta} \right)}{n} \right) \tag{24}$$

  where K is an appropriate numeric constant, $\mathcal{R}$ is the Radamacher complexity of the space of functions $\mathcal{F}$.

  Under the same hypotheses that guarantee compactness for the space of solution of SGD, $\mathcal{R}^2$ goes to zero with increasing N because the covering numbers provides an upper bound on Radamacher averages. Additionally, by checking Equation 24, we can observe that zero-minimizers have a better generalization bound if they have larger margin.

Theory I results (see 8.1) imply that $\mathcal{N}(F)$ for d-dimensional functions of a Sobolev space with smoothness s grow as $\log \mathcal{N}(\mathcal{F}) \sim (\frac{1}{\epsilon})^{\frac{d}{s}}$. On the other hand for the subset of hierarchically local compositional functions ($\mathcal{F}_c$ – *and for the set of functions generated by the corresponding networks of which convolutional networks are a subset* – the covering numbers are much smaller, only growing as $\log \mathcal{N}(\mathcal{F}_c) \sim d(\frac{1}{\epsilon})^{\frac{h}{s}}$ as a function of d.

Since

$$\mathcal{R}_n \leq C \cdot \inf_{\epsilon > 0} \left\{ \epsilon + \frac{1}{\sqrt{n}} \int_{\epsilon}^{\|\mathcal{F}\|_\infty} \sqrt{\log \mathcal{N}_\delta(\mathcal{F})} d\delta \right\} \tag{25}$$

we conclude that the term $\mathcal{R}_n^2$ is much smaller for convolutional networks than for equivalent shallow networks. This may be the reason for the good prediction properties of deep networks when the underlying function to be learned is compositional. Furthermore, the covering numbers of the subset of functions in a ball around the golbal minimizer should be even smaller, because the effective d is dictated by the number of data and not by the much larger number of weights. Apart from this non-rigorous argument,, generalization is also controlled by the margin γ. We now collect together the results from Theory I and II and from the previous sections in this paper to claim the following

Proposition 1.
- *Good expected error by deep networks relative to shallow networks in tasks that correspond to learn a compositional function (or a linear combination of a "small" number of them) follows from the smaller Radamacher complexity .*
- *Consistency is further controlled by the empirical margin of minimizers that depends not only on the distributions of inputs x_i but also on the distribution of y_i.*
- *Later we will show that SGDL prefers among the minima the degenerate zero-minimizers – if they exist; among the degerate zero-minimizers it prefers the ones with flattest minima, that, as we will show later, correspond to the largest margin.*

In summary, SGDL (and likely also SGD) selects with high probability the minimizers that do the best job in terms of minimizing expected error. This suggests that the large-margin argument (see also later) to explain consistencyin the regime $W > N$ also explains generalization in the regime $W < N$.

8.5 Global minimization and stability under regularization-like assumptions

The most interesting proofs for our goal are available for the case of Langevin equations and annealing techniques where results about global optimization and stability of the algorithm have become available. Gelfand and Mitter [24, 25] prove convergence of Langevin equations to global minima. Related results on simulated annealing for finite Markov chains ([26]) prove polynomial convergence time (meaning probability of visiting the global optimum at least once). A very recent paper by Raginsky et al.[27] considers a version of SGD – called SGDL – in which isotropic Gaussian noise is added to an estimate of the gradient. They also prove global convergence but in addition the prove generalization with rates independent of parameter numbers or size and valid in the case of general loss functions. In order to apply their results to standard SGD one should first prove that SGD is a good approximation of a Langevin equation (and that their results still hold approximatively). We skip this step because there are empirical reasons to believe that SGDL may the better way to train neural networks. In particular our numerical analysis suggests that the behavior of SGDL is very similar to SGD but SGDL consistently shows a slightly better generalization performance. In the next section, hoever, we provide theoretical and numerical arguments for the intuition that SGD is similar to SGDL and in fact to a Langevin equation (GDL) . This claim is not new: there are previous analyses of it. As an example see[28].

9 Second Appendix: Various Topics

9.1 Generalization Bounds: comparison with compositional networks

The following is from section 8 in Theory I [1], provided here for completeness.

Assume a network size that ensure the same approximation error ϵ . Then in order to achieve the same generalization error ϵ_G, the sample size $M_{shallow}$ of the shallow network must be much larger than the sample size M_{deep} of the deep network:

$$\frac{M_{deep}}{M_{shallow}} \approx \epsilon^n. \tag{26}$$

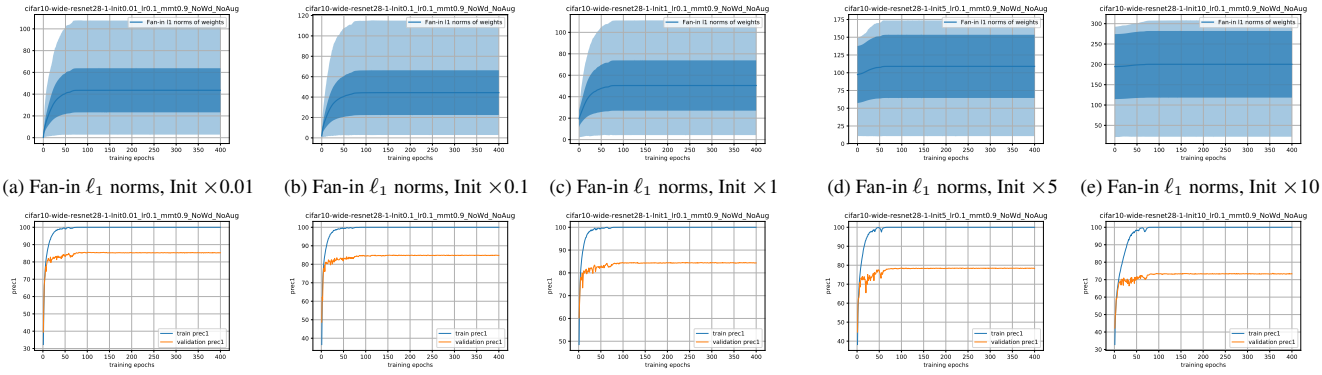
This implies that for largish N there is a (large) range of training set sizes between M_{deep} and $M_{shallow}$ for which deep networks will not overfit (corresponding to small ϵ_G) but shallow networks will (for dimensionality $N \approx 10^4$ and $\epsilon \approx 0.1$ Equation 26 yields $m_{shallow} \approx 10^{10^4} m_{deep}$).

A similar comparison is derived if one considers the best possible expected error obtained by a deep and a shallow network. Such an error is obtained finding the architecture with the best trade-off between the approximation and the estimation error. The latter is essentially of the same order as the generalization bound implied by inequality (14), and is essentially the same for deep and shallow networks, that is

$$\frac{rn}{\sqrt{M}}, \tag{27}$$

where we denoted by M the number of samples. For shallow networks, the number of parameters corresponds to r units of N dimensional vectors (plus off-sets), whereas for deep compositional networks the number of parameters corresponds to r units of 2 dimensional vectors (plus off-sets) in each of the $N-1$ units. Using our previous results on degree of approximation, the number of units giving the best approximation/estimation trade-off is

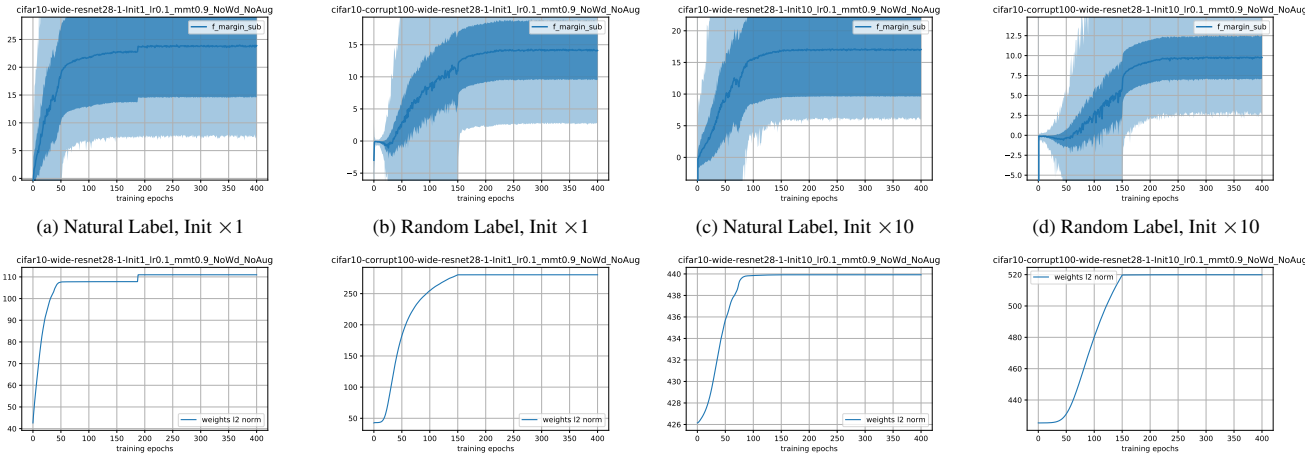
$$r_{shallow} \approx \left(\frac{\sqrt{M}}{n}\right)^{\frac{n}{m+n}} \qquad \text{and} \qquad r_{deep} \approx \left(\sqrt{M}\right)^{\frac{2}{m+2}} \tag{28}$$



(a) Fan-in ℓ_1 norms, Init $\times 0.01$ (b) Fan-in ℓ_1 norms, Init $\times 0.1$ (c) Fan-in ℓ_1 norms, Init $\times 1$ (d) Fan-in ℓ_1 norms, Init $\times 5$ (e) Fan-in ℓ_1 norms, Init $\times 10$

(f) Learning curves, Init $\times 0.01$ (g) Learning curves, Init $\times 0.1$ (h) Learning curves, Init $\times 1$ (i) Learning curves, Init $\times 5$ (j) Learning curves, Init $\times 10$

Figure 16: Fan-in ℓ_1 norms of the weights for training on CIFAR-10.



(a) Natural Label, Init $\times 1$ (b) Random Label, Init $\times 1$ (c) Natural Label, Init $\times 10$ (d) Random Label, Init $\times 10$

(e) Natural Label, Init $\times 1$ (f) Random Label, Init $\times 1$ (g) Natural Label, Init $\times 10$ (h) Random Label, Init $\times 10$

Figure 17: Functional margins for training on CIFAR-10. Patterns: random labels has smaller margin and larger l2 norm than random labels. Larger init ($\times 10$) leads to smaller margin and larger l2 norm than normal init ($\times 1$).

for shallow and deep networks, respectively. The corresponding (excess) expected errors $\mathcal{E}$ are

$$\mathcal{E}_{shallow} \approx \left(\frac{n}{\sqrt{M}}\right)^{\frac{m}{m+n}} \tag{29}$$

for shallow networks and

$$\mathcal{E}_{deep} \approx \left(\frac{1}{\sqrt{M}}\right)^{\frac{m}{m+2}} \tag{30}$$

for deep networks. For the expected error, as for the generalization error, deep networks appear to achieve an exponential gain. The above observations hold under the assumption that the optimization process during training finds the optimum parameters values for both deep and shallow networks. Taking into account optimization, e.g. by stochastic gradient descent, requires considering a further error term, but we expect that the overall conclusions about generalization properties for deep vs. shallow networks should still hold true.

Notice that independently of considerations of generalization, deep compositional networks are expected to be *very efficient memories* – in the spirit of hierarchical vector quantization – for associative memories reflecting compositional rules (see Appendix and [29]). Notice that the advantage with respect to shallow networks from the point of view of memory capacity can be exponential (as in the example after Equation 26 showing $m_{shallow} \approx 10^{10^4} m_{deep}$).

9.2 Empirical results on margin

The key question however remains: why does the combination of SGD and Deep Convolutional Network work so well? Why is the margin so good? Our argument in this section is that among the solutions that minimize the empirical error, SGDL selects those that are as robust as

possible. The next step in the argument is that robust minima are flat in weights space. The last step is to show that flatness is equivalent to large margin.

9.3 SGD is equivalent to a form of Robust Optimization and corresponds to large margin

The claim is that SGD while minimizing the empirical error effectively maximizes robustness to perturbations in the weights (and similarly in the training data). In particular, section 9.4 shows that this is similar to regularization with a regularizer consisting of a path norm in the weights of the deep network. In the subsequent section 8.3 we outline results – old and new – showing that robustness yield generalization bounds that are independent of number of parameters. Unfortunately these bounds are very loose.

In the following sections we argue that flatness in the minimum provided by SGD yields better generalization. We do not have a practical bound but we present strong arguments based on a a close connection between flatness, robust optimization and regularization.

Let us first introduce a notation to describe the deep networks that we will use in the following sections.

Netwoks and weights

We assume that f is a L-layer neural network with d inputs, trained on $z_1, ...z_d$ with $z = (x, y)$. We assume one of the components of the effective input vector x is a dummy input equal to 1 (this allows to drop the parameter b in describing the RELU activity). For convenience we repeat the definitions in (20), (21), and (22). The input to the first layer is

$$x^0 := x;$$

the output of layer $v-1$ is x^v with components

$$\forall v = 1, \cdots, L-1 \colon x_i^v := [\sum_{j=1}^{N_{v-1}} w_{i,,j}^{v-1} x_j^{v-1}]_+ i = 1, \cdots, d_v$$

and

$$f(x) = [\sum_{j=1}^{N_{L-1}} w_j^{L-1} x_j^{L-1}]_+.$$

Notice that if we neglect the RELUs (or equivalently assume that all the RELUs are active), the L layer network with d inputs and one output, is equivalent to a linear network with a d-dimensional weight vector $\tilde{\mathbf{w}} = \sum_{h,t,j\cdots} w_{1,h}^{L-1} \cdots w_{t,k}^2 w_{k,j}^1 w_{j,i}^0$ with $i = 1, \cdots, d$. Notice that $||\tilde{\mathbf{w}}||$ is the path norm defined elsewhere [30]. The vector $\tilde{\mathbf{w}}$ involved in the path norm has components which are each a sum of monomials in the weights, each monomial corresponding to a path. Suppose there are a total of D paths. We define $\hat{\mathbf{w}}$ as the D-dimensional vector in which each component consist of the monomial corresponding to a path and as $\hat{\mathbf{x}}$ the corresponding "augmented" x vector (for instance in Figure 18 $D = d$ if we assume one RELU per node; with 2 RELUs per node $D = 4d$). The corresponding "Dpath norm" is $||\hat{\mathbf{w}}||$. Thus

$$f(x) = \Lambda_{w,x} \hat{\mathbf{w}} \hat{\mathbf{x}} \tag{31}$$

where Λ is a diagonal matrix with $0, 1$ components indication which paths are switched on. Notice that $\frac{df}{dw} \approx 0$ around a flat minimum w^* of f.

9.4 SGDL, RO and generalization

In the next subsection we extend previous results [31] to relate minimizers that corresponds to flatness in the minimum to robustness: thus SGD performs robust optimization of a certain type. In particular, we will show that robustness to perturbations in multilayer deep networks is related to regularization (see [31, 32]). We first describe the simplest separable and linear case.

9.4.1 For linear networks and separable data, SGD converges to a large-margin classifier

In linear classification we try to separate the two classes of a binary classification problem by a hyperplane $\mathcal{H} = \{x : w^\intercal x + b = 0\}$, where $w \in \mathbb{R}^d$. There is a corresponding decision rule of the form $y = sign(w^\intercal x + b)$ Let us consider the separable condition in which the decision rule makes no error on the data set. This corresponds to the following set of inequalities

$$y_i(w^\intercal x_i + b) > 0, \quad i = 1, \cdots, n \tag{32}$$

We assume that the inequalities are feasible, that is the data are separable. Assume now to impose robustness of the classifier wrt to perturbations δw in the weights which we assume here to be such that $||\delta w||_2 \leq \rho||w||_2$ with $\rho \geq 0$. In other words, we require that $\forall \delta w$ such that $\|\delta w\|_2 \leq \rho\|w\|_2$:

$$y_i((w + \delta w)^\intercal x_i + b) \geq 0, \quad i = 1, \ldots, n \tag{33}$$

Further assume that $\|x_i\|_2 \approx 1$, then for $i = 1, \ldots, n$, if we let $\delta w = -\rho y_i x_i \|w\|_2 / \|x_i\|_2$,

$$y_i(w^\intercal x_i + b) \geq -y_i \delta w^\intercal x_i = \rho\|w\|_2\|x_i\|_2 \approx \rho\|w\|_2$$

As a result, an approximate *robust counterpart* of Equation (32) is

$$y_i(w^\intercal x_i + b) \geq \rho||w||_2, \quad i = 1, \cdots, n \tag{34}$$

Maximizing ρ – the margin – subject to the constraints Equation (34) leads to minimizing w, because we can always enforce $\rho||w||_2 = 1$ and thus to

$$\min_{w,b}\{||w||_2 : y_i(w^\intercal x_i + b) \geq 1 \quad i = 1, \cdots, n\} \tag{35}$$

Notice that the same Equation 35 follows if the maximization is with respect to spherical perturbations of radius ρ around each data point x_i. In either case the resulting optimization problems is equivalent to hard margin SVMs[4]. Notice that the classification problem is similar to using the loss function $V(y, f(x)) = \log(1 + e^{-yf(x)})$ which penalizes errors but is otherwise very small in the zero classification error case. Section 5.2 implies that SGD maximizes flatness at the minimum that is SGD maximizes δw.

In the non-separable case, the hinge loss[5] leads to the following robust minimization problem

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(w^\intercal x_i + b) + \rho||w||_2]_+. \tag{37}$$

Note that the robust version of this worst-case loss minimization is not the same as in classical SVM because the regularization term is inside the hinge loss. Note also that standard regularization is an upper bound for the robust minimum since

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(w^\intercal x_i + b) + \rho||w||_2]_+ \leq \min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(w^\intercal x_i + b)]_+ + \rho||w||_2]. \tag{38}$$

In the case of the square loss, robust optimization gives with $||\delta w||_2 \leq \rho||w||_2$

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [(y_i - w^\intercal x_i)^2 + \rho^2||w||_2]_+. \tag{39}$$

The summary here is that depending on the loss function and on the uncertainity set allowed for the perturbations δw one obtains a variety of robust optimization problems. In general they are not identical to standard regularization but usually they contain a regularization term. Notice that a variety of regularization norms (for instance ℓ_1 and ℓ_2) guarantee CV_{loo} stability and therefore generalization. The conclusion is that *in the case of linear networks and linearly separable data, SGD provides a solution closely related to hinge-loss SVM.*

[4] If we start from the less natural assumption that $||\delta w||_\infty \leq \rho||w||_2$ with $\rho \geq 0$ and assume that each of the d components $|(x_i)_j| \approx 1, \quad \forall i$d, then the *robust counterpart* of Equation 32 is, since $(\delta w)^\intercal x \leq ||(\delta w)^\intercal||_{\ell_\infty}||x||_{\ell_1}$,

$$y_i(w^\intercal x_i + b) \geq (\delta w)^\intercal \sup \delta w = \rho||w||_2, \quad i = 1, \cdots, n \tag{36}$$

[5] The hinge loss $V(y, f(x)) = [1 - yf(x)]_+$, with y binary (used by [32]) is similar to the cross-entropy loss (for classification) $V(y, f(x)) = \log(1 + e^{-yf(x)})$.

9.4.2 Deep networks, SGD, RO and regularization

In the case of deep (convolutional) networks we can *only approximate* the robust optimization problems described above *at a minimum* by using the pseudo-linear representation of a deep networks provided by Equation 31, where Λ depends on the weights and on the input x. For a deep network the problem around the minimum for the square loss under the assumptions of the previous section reads

$$\min_{w} \max_{(\delta w)} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_w(\mathbf{x_i})^2 \tag{40}$$

which, since $f(x) = \Lambda_{w,x} \hat{\mathbf{w}} \hat{\mathbf{x}}$, leads to

$$\min_{\hat{\mathbf{w}}} \frac{1}{n} \sum_{i=1}^{n} [(y_i - \hat{\mathbf{w}}^\intercal \Lambda_{w,x} \hat{\mathbf{x_i}})^2 + \frac{1}{n} \rho^2 \hat{\mathbf{w}}^\intercal \Lambda_{w,x} \hat{\mathbf{w}}. \tag{41}$$

Here the diagonal matrix Λ weighs each path by the times it is active over the training set S_n. Of course the perturbation will switch on or off several of the paths, switching on or off associated RELUs, depending on the training sample $y_i, (x_i$. We consider the simple case of Figure 18, where the number of paths is equal to the dimensionality of the input vector. Let us assume that there are several zero minima in the square loss, all close to zero loss. Then maximization over δ selects the flattest minima – the ones where perturbation can be largest without increasing the errors over the training set. Let us assume as before that all the explored minima are at close to zero loss, that is $y_i - f(x_i) \approx 0$. In this case the robust optimum is close to the minimum of a weighted regularization problem.

In fact the penalization term above is an upper bound so that

$$\min_{w} \max_{(\delta w)} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_{w+\delta w}(\mathbf{x_i})^2 \leq \min_{\hat{\mathbf{w}}} \frac{1}{n} \sum_{i=1}^{n} [(y_i - \hat{\mathbf{w}}^\intercal \Lambda_{w,x} \hat{\mathbf{x_i}})^2 + \frac{1}{n} \rho^2 \hat{\mathbf{w}}^\intercal \Lambda_{w,x} \hat{\mathbf{w}}. \tag{42}$$

More in general

Proposition 2. *For a compositional network with several units per node the following holds*

$$\min_{\hat{\mathbf{w}}} \frac{1}{n} \sum_{i=1}^{n} [(y_i - \hat{\mathbf{w}}^\intercal \Lambda_{w,x} \hat{\mathbf{x_i}})^2 + \frac{1}{n} \rho^2 \hat{\mathbf{w}}^\intercal \hat{\mathbf{w}} \leq \min_{w} \max_{\delta \in \mathcal{N}} \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(\mathbf{x_i} - \delta)) \leq \min_{\hat{\mathbf{w}}} \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(\mathbf{x_i})) + c|\hat{\mathbf{w}}| \leq \min_{\hat{\mathbf{w}}} \sum_{i=1}^{n} [(y_i - \hat{\mathbf{w}}^\intercal \Lambda_{w,x} \hat{\mathbf{x_i}})^2 + \rho^2 \hat{\mathbf{w}}^\intercal \Lambda_{w} \tag{43}$$

where the lower and upper bounds depend on the assumption that each path is active at least once over the training set.

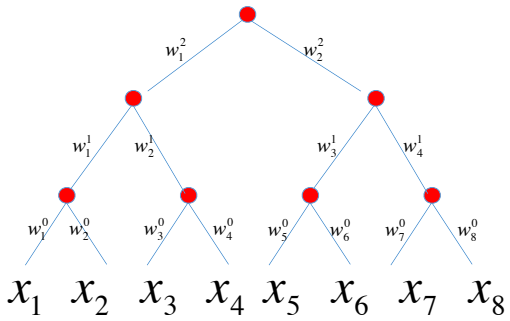
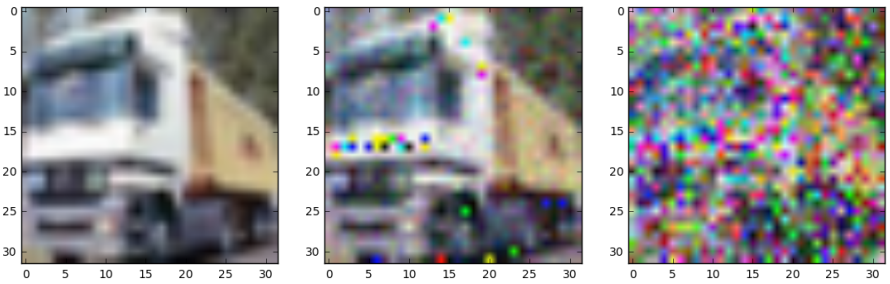


Figure 18: Assume that the binary network in the figure has one relu unit per node. The path norm is $|\hat{\mathbf{w}}|$ and $\hat{\mathbf{w}} = w_1^0 w_1^1 w_1^2, w_2^0 w_1^1 w_1^2, w_3^0 w_2^1 w_1^2, \cdots$. If one of the unit is switched off then the two weights in the input to the unit can be put to zero and one or more of the monomials representing the components of $\hat{\mathbf{w}}$ will be zero. In the case of several units per node, the components of $\hat{\mathbf{w}}$ will be sum of monomials, some of which can be put to zero if corresponding RELU units are switched off.

It seems likely that we can bound CV_{loo} stability (for any N) and guarantee generalization. This may require to make ρ decrease with N by controlling the power of a Gaussian noise added to SGD (we assume to be in the SGDL case). In summary we can establish a close connection between the flatness maximized by SGD, maximizing robustness and regularization *but only* approximately and only in some special cases.

Remarks



(a) Original (b) Perturbation 0.05 (c) Perturbation 0.2

Figure 19: Example of an image on the CIFAR-10 training set and its perturbed versions with different perturbation size.

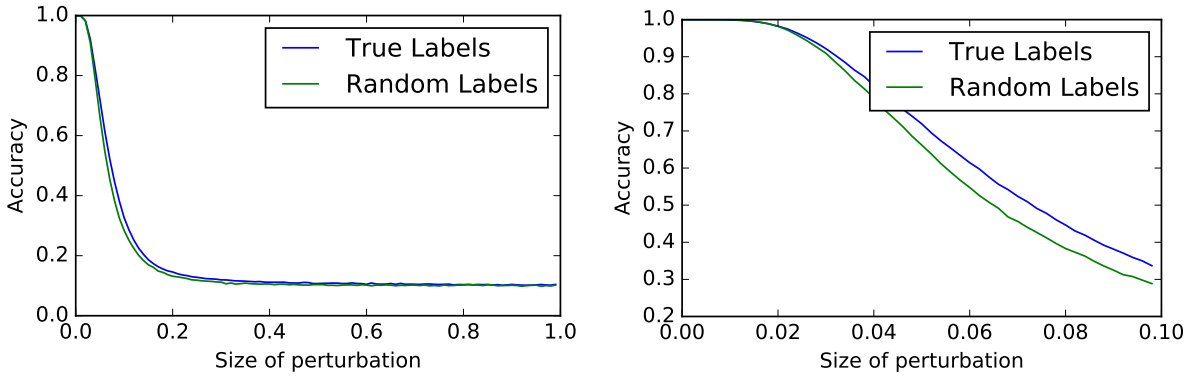


Figure 20: Perturbation robustness of models trained by SGD on CiFAR with correct and random labels and then tested on perturbed images. The right hand side figure is a zoomed-in view of the initial part of the left hand side figure.

- The proposition suggests that a deep network optimized by SGD has bounded weights, uniform stability and generalization. The associate bounds are *independent* of the number of parameters.

9.5 Training with random labels, testing with perturbed images

In this section, we empirically compare the robustness of the classifier learned on natural labels and random labels. Specifically, since for both natural labels and random labels, the models could perfectly fit the labels, we are interested in comparing how the fitted model respond to small perturbations on the training examples. We define the following notion of perturbation robustness:

$$\tilde{E}(f, S_n, \gamma) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[f(x_i + \gamma\varepsilon_i) \neq y_i] \tag{44}$$

where $S_n = \{(x_i, y_i)\}_{i=1}^n$ is the training set, f is a model that was trained to fit the training set. $\varepsilon_1, \ldots, \varepsilon_n$ are i.i.d. standard Gaussian vectors. And γ is the size of the perturbation. When the model fits the training set perfectly, $\tilde{E}(f, S_n, 0) = 0$.

Figure 19 shows an example of the training image and its perturbed version with different perturbation sizes. Figure 20 shows the perturbation robustness with models trained on natural labels and random labels. As we can see, the model trained on random labels has slightly worse robustness than the model trained on natural labels.

9.6 GD and SGD converges to Minimum Norm Solution for Underdetermined Linear System

Consider the following setup: $X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n\times d}$ are the data points, with $d > n$. We further assume that the data matrix is of full row rank: $\text{rank}(X) = n$. Let $y \in \mathbb{R}^n$ be the labels, and consider the following linear system:

$$Xw = y \tag{45}$$

where $w \in \mathbb{R}^d$ is the weights to find. This linear system has infinite many solutions because X is of full row rank and we have more parameters than the number of equations. Now suppose we solve the linear system via a least square formulation

$$L(w) = \frac{1}{2n}\|Xw - y\|^2 \tag{46}$$

by using gradient descent (GD) or stochastic gradient descent (SGD). In this section, we will show that both GD and SGD converges to the minimum norm solution, for the ℓ_2 norm. A similar analysis (with fewer details) was presented in [4].

Lemma 2. *The following formula defines a solution to* (45)

$$w_\dagger \triangleq X^\top (XX^\top)^{-1} y \tag{47}$$

and it is the minimum norm solution.

Proof Note since X is of full row rank, so $XX^\top$ is invertible. By definition,

$$Xw_\dagger = XX^\top (XX^\top)^{-1} y = y$$

Therefore, $w_\dagger$ is a solution. Now assume $\hat{w}$ is another solution to (45), we show that $\|\hat{w}\| \geq \|w_\dagger\|$. Consider the inner product

$$\begin{aligned}
\langle w_\dagger, \hat{w} - w_\dagger \rangle &= \langle X^\top (XX^\top)^{-1} y, \hat{w} - w_\dagger \rangle \\
&= \langle (XX^\top)^{-1} y, X\hat{w} - X_\dagger \rangle \\
&= \langle (XX^\top)^{-1} y, y - y \rangle \\
&= 0
\end{aligned}$$

Therefore, $w_\dagger$ is orthogonal to $\hat{w} - w_\dagger$. As a result, by Pythagorean theorem,

$$\|\hat{w}\|^2 = \|(\hat{w} - w_\dagger) + w_\dagger\|^2 = \|\hat{w} - w_\dagger\|^2 + \|w_\dagger\|^2 \geq \|w_\dagger\|^2$$

Lemma 3. *When initializing at zero, the solutions found by both GD and SGD for problem* (46) *live in the span of rows of X. In other words, the solutions are of the following parametric form*

$$w = X^\top \alpha \tag{48}$$

for some $\alpha \in \mathbb{R}^n$.

Proof

The gradient for (46) is

$$\nabla_w L(w) = \frac{1}{n} X^\top (Xw - y) = X^\top e$$

where we define $e = (1/n)(Xw - y)$ to be the error vector. GD use the following update rule:

$$w_{t+1} = w_t - \eta_t \nabla_w L(w_t) = w_t - \eta_t X^\top e_t$$

Expanding recursively, and assume $w_0 = 0$. we get

$$w_t = \sum_{\tau=0}^{t-1} -\eta_\tau X^\top e_\tau = X^\top \left(-\sum_{\tau=0}^{t-1} \eta_\tau e_\tau \right)$$

The same conclusion holds for SGD, where the update rule could be explicitly written as

$$w_{t+1} = w_t - \eta_t (x_{i_t}^\top w - y_{i_t}) x_{i_t}$$

where (x_{i_t}, y_{i_t}) is the pair of sample chosen at the t-th iteration. The same conclusion follows with $w_0 = 0$.

Q.E.D.

Theorem 4. *Let w_t be the solution of GD after t-th iteration, and w_t be the (random) solution of SGD after t-th iteration. Then $\forall \varepsilon > 0$, $\exists T > 0$ such that*

$$L(w_t) \leq \varepsilon, \quad \mathbb{E} L(\mathsf{w}_t) \leq \varepsilon$$

where the expectation is with respect to the randomness in SGD.

Corollary 1. *When initialize with zero, both GD and SGD converges to the minimum-norm solution.*

Proof Combining Lemma 3 and Theorem 4, GD is converging $w_t \to w_\star = X^\top \alpha_\star$ as $t \to \infty$ for some optimal $\alpha_\star$. Since $w_\star$ is a solution to (45), we get

$$y = Xw_\star = XX^\top \alpha_\star$$

Since $XX^\top$ is invertible, we can solve for $\alpha_\star$ and get

$$w_\star = X^\top \alpha_\star = X^\top (XX^\top) y = w_\dagger$$

Similar argument can be made for SGD with expectation with respect to the randomness of the algorithm.

9.6.1 Weight Perturbations

We could define in different ways the basic perturbations of the data that we consider [6] but here we assume independent perturbation of each component of a vector of perturbations $\delta_i = \delta$ which is the same for each example x_i.

Suppose that there is zero-minimizer $f*$ of the empirical loss with parameters w^*, yielding $I_{S_n}(w^*) = 0$. Suppose further that the minimizer is Δw-*flat*, e.g. $I_{S_n}(w^* + \Delta w) = 0$, where Δw is a vector of perturbations of the weights w^* and $\min |\Delta w_i| \geq C$.

For an illustration of the basic idea, consider just the first layer with $P = 1$. Then calling $x_j^0 = x_j$ we obtain with Δ being a percentage perturbation of w

$$x_i^1 = \sum_{j=1}^{2} (w_{i,j}^0 \pm \Delta w_j) x_j = \sum_{j=1}^{2} (w_{i,j}^0 (x_j \pm \Delta x_j) \tag{49}$$

implying that a delta percentage of weight perturbation corresponds to the same percentage of x-data perturbation, that is to Δ-robustness.

Consider now the case of a multilayer network with a graph corresponding to a binary tree and $P = 1$. Then a "flatness" $\pm\Delta$ in all weights – we now consider a fraction Δ of the maximum range of the weights (which are between π and $-\pi$) – corresponds to the following maximum perturbation in the contribution of component x_j to the output (when it is not zero because of ReLU)

$$[\prod_{d=1}^{L} (1 \pm (\Delta) w_j^d)] x_j. \tag{50}$$

Equation 50 has to be compared with the effect of perturbed data

$$[\prod_{d=1}^{L} w_j^d](1 \pm \Delta) x_j. \tag{51}$$

This suggests the following

Conjecture 3. *Invariance of the empirical error to perturbations of all weights are equivalent to larger, identical perturbations of all training examples x_i with $i = 1, \cdots, n$.*

As an example, supposed there are $d = 8$ layers and that $\Delta = \frac{1}{10}$. Then this tolerance in all the weights corresponds to a tolerance in the data perturbation of a factor 2 (1.1 vs 2.14). In addition, note also that there are usually many more weights than data. This relatively small δ "flatness" of the zero-minimizer in weight space yields large robustness in data space (in this picture, to the same perturbations for all data points). The product of the weights that appears in the equations above is related to the path norm that we will discuss later (see also Figure 18).

9.7 Stability of SGD

In this section we discuss the equivalence of the following properties (some have appeared in the literature):

- *flat minima in the weights of the minimizer*
- *stability of the minimizer wrt perturbations of weights*
- *stability wrt perturbations of the data*
- *fewer bits needed for parameters*

[6]

Definition 4. *A set $\mathcal{N}_0 \in \mathcal{R}^n$ is called an Atomic Uncertainity Set if*

1. $\mathbf{0} \in \mathcal{N}_0$
2. *For any* $\mathbf{w}_0 \in \mathcal{R}^n$ $\sup_{\delta \in \mathcal{N}_0} [\mathbf{w}_o^\intercal \delta] = \sup_{\delta' \in \mathcal{N}_0} [-\mathbf{w}_0 \delta'] < \infty$

Based on this [32] then define more complex models of perturbations in which the behavior across multiple samples $i = 1, \cdots, n$ is controlled.

- *"small" local complexity (Ramacher averages) of functions that are flat*

Stability plays an important role in several different situations such as in inverse problems, in dynamical systems – under the form of "structural stability" – and in learning – as stability under perturbations of the training set. The intuition is that these are very similar forms of stability but formally the different cases requires different definitions and their mathematical equivalence remains in general elusive.

We discuss here at the intuitive level the relation between different types of perturbations in the specific case of SGD, that we define as before as

$$f_{t+1} = f_t - \gamma_n \nabla V(f_t, z_t), \tag{52}$$

- *SGD can be written as Gradient Descent with additive noise*
  We rewrite
  $$\nabla V(f_t, z_t) = \nabla I_{S_n}(f_t) + \xi_t \tag{53}$$

  As we discussed earlier, the SGD update step is then rewritten as the following Langevin equation
  $$f_{t+1} = f_t - \gamma_n(\nabla I_{S_n}(f_t) + \xi_t) \tag{54}$$

- *Perturbing the training examples in SGD – which we described in the context of perturbation stability or robustness – is equivalent to a pointwise change in the noise process in the equivalent noisy GD* Suppose that z_j is replaced by z_j' for a given j in S_n. Then $U(f_t)$ is replaced by $U'(f_t) = U(f_t) - V(f_t, z_j) + V(f_t, z_j') = U(f_t) + \xi_j'$ where ξ_j' is a "noise" term.

- *Perturbing the hypothesis f_t, that is the weights, is equivalent to adding an additional noise term to noisy GD*
  Suppose the parameters w are by replacing them component-wise with $w + \tau w$. Then $\nabla_w U(f_f) \to \nabla_w U(f_f)(I + \tau) = \nabla_w U(f_f) + \xi$".

The three cases end up being equivalent to noisy GD with slightly different noise terms. Notice that GD and SGD converge to a minimum norm solution in the linear degenerate case.

We conjecture that under some assumptions each of the cases above can be transformed in another one. This means that GD with replacement can be seen as noisy GD and this in turn may be seen under appropriate conditions as GD with replacement. The relevance of this observation is that stability wrt data is thus ensured by the presence of some form of noise in GD such as provided by the SGD update.

Notice that convergence to the same minima or to a class of equivalence is a form of *structural stability of the dynamical system* associated with SGD. Our intuition is that "repeat SGD" generalizes because the "noisy" update of the gradient makes it equivalent to "pure SGD" for $N \to \infty$ in S_n.

- The key to generalization by "repeat SGD" is the intrinsic noise associated with SGD. This is different wrt GD. Because of properties of the Boltzman distribution in high dimensions, SGD finds with high probability *structurally stable* flat zeros (that is robust, wrt to perturbations of the weights). These coincide with stable solutions that generalize.
- The relative amount of noise can be controlled by changing the size of the minibatches.
- SGD solutions generalize because each example z_n with its repetitions (multiple passes) provides effectively different data points because of the noise term. This is formally correct if we could prove that perturbing the sample z_n is equivalent to a change in the noise term which does not affect the properties of SGD and associated theorems. Then the pure SGD theorem applies and convergence to the expected risk is guaranteed.
- Another intuition is that the solutions found by SGD are not just zeros of the polynomial set of equations of Theory II but "stable zeros" because of the noise and the fact that SGD is a Langevin equation converging in probability to the minima of the cost functions and preferring the stable ones.

9.8 Compositionality Improves Generalization Bounds

It has been long recognized that capacity of the set of neural networks can be controlled not only through the number of units (and, hence, the VC dimension) but also through the size of the weights [33]. In particular, the argument of [34] leads to a bound on the generalization error in terms of ℓ_1-norms of the weights into each neuron. If each weight in the network is bounded, the result yields a non-vacuous generalization guarantee for any network with number of incoming connections into each neuron being $o(n)$, the sample size. For completeness, we provide the statement of [34] here, with minor modifications.

Define

$$\mathcal{F}_1 = \left\{x \mapsto \langle w^1, x\rangle : \|w^1\|_1 \leq B_1\right\} \tag{55}$$

and

$$\mathcal{F}_i = \left\{x \mapsto \sum_j w_j^i \sigma(f_j(x)) : f_j \in \mathcal{F}_{i-1}, \|w^i\|_1 \leq B_i\right\} \tag{56}$$

for $i \geq 2$, and suppose $x \in \mathcal{X} \subset [-1,1]^d$.

Lemma 4. *Consider a k-layer neural network defined recursively as in* (56)*. Let $V : \mathbb{R} \times \{\pm 1\} \to \mathbb{R}$ be 1-Lipschitz loss function, and write $V(f,z) = V(f(x),y)$ for $z = (x,y) \in \mathcal{X} \times \{\pm 1\}$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be 1-Lipschitz and $\sigma(0) = 0$.*[7] *Then*

$$\mathbb{E} \sup_{f \in \mathcal{F}_k} \left\{\mathbb{E}V(f,Z) - \frac{1}{n}\sum_{t=1}^n V(f,Z_t)\right\} \leq 2\mathbb{E}\widehat{\mathcal{R}}(\mathcal{F}_k) \leq \frac{\sqrt{2\log d}\cdot \prod_{i=1}^k (2B_i)}{\sqrt{n}}, \tag{57}$$

where $\widehat{\mathcal{R}}(\mathcal{F}_k)$ are empirical Rademacher averages of $\mathcal{F}_k$.

Proof. Using the standard symmetrization argument followed by contraction, the uniform deviations in (57)

$$2\mathbb{E} \sup_{f \in \mathcal{F}_k} \frac{1}{n}\sum_{t=1}^n \epsilon_t V(f,Z_t) \leq 2\mathbb{E} \sup_{f \in \mathcal{F}_k} \frac{1}{n}\sum_{t=1}^n \epsilon_t f(X_t). \tag{58}$$

Now, for any $i \geq 2$,

$$\widehat{\mathcal{R}}(\mathcal{F}_i) \quad = \quad \mathbb{E} \sup_{f_j \in \mathcal{F}_{i-1}, \|w^i\| \leq B_i} \frac{1}{n}\sum_{t=1}^n \epsilon_t \sum_j w_j^i \sigma(f_j(X_t)) \leq 2B_i \widehat{\mathcal{R}}(\mathcal{F}_{i-1}) \tag{59}$$

where the last step uses the Cauchy-Schwartz inequality. Finally,

$$\widehat{\mathcal{R}}(\mathcal{F}_1) = B_1 \left\| \frac{1}{n}\sum_{t=1}^n \epsilon_t X_t \right\|_\infty \leq B_1 \sqrt{\frac{2\log d}{n}}$$

using the usual estimates. □

Lemma 4, together with the empirical observation that neural networks can be trained to achieve zero error on the data, implies that the expected risk of the minimizer is converging to zero. Lemma 4 assumes that V is Lipschitz. However, the standard argument from the theory of large margin classifiers yields a guarantee on the expected zero-one loss as well, in terms of the Rademacher averages of the class and the empirical margin. This result, which can be found in [22, Theorem 2] (see also [35] for the present version), is stated below:

Theorem 5. *For all $\delta > 0$, with probability at least $1-\delta$, for all $f \in \mathcal{F}_k$ and $\gamma > 0$,*

$$P(Yf(X) \leq 0) \leq \frac{card(i : Y_i f(X_i) < \gamma)}{n} + \frac{4}{\gamma}\mathbb{E}\widehat{\mathcal{R}}(\mathcal{F}_k) + \sqrt{\frac{\log\log(4C/\gamma)}{n}} + \sqrt{\frac{\log \delta^{-1}}{2n}}, \tag{60}$$

where $C \geq \sup_{f,x} f(x)$.

The bound of (4) on Rademacher averages can be used in Theorem 5 to obtain an upper bound on the expected zero-one loss of a solution that has a large margin γ on the data. Notice that γ optimizing the above bound is based on the empirical margin for the particular draw of the data.

We observe that the covering numbers are much better for compositional networks than for densely connected networks consistently with section 9.1; the covering number are an upper bound to the Radamacher averages.

9.9 Compositionality and Multiclass Tasks

Most of the succesfull neural networks exploit compositionality not only to avoid the curse of dimensionality but also for better generalization in an important way (see [36]). Suppose that the mappings to be learned in a family of classification tasks (for instance classification of different object classes in Imagenet) may be approximated by compositional functions such as $f(x_1,\cdots,x_n) = h_l \cdots (h_{21}(h_{11}(x_1,x_2),h_{12}(x_3,x_4)),h_{22}(h_{13}(x_5,x_6),h_{14}(x_7,x_8)\cdots))\cdots)$, where h_l depends on the task (for instance to which object class) but all the other constituent functions h are common across the tasks. Under such an assumption, multi-task learning, that is training

[7] This assumption can be easily removed.

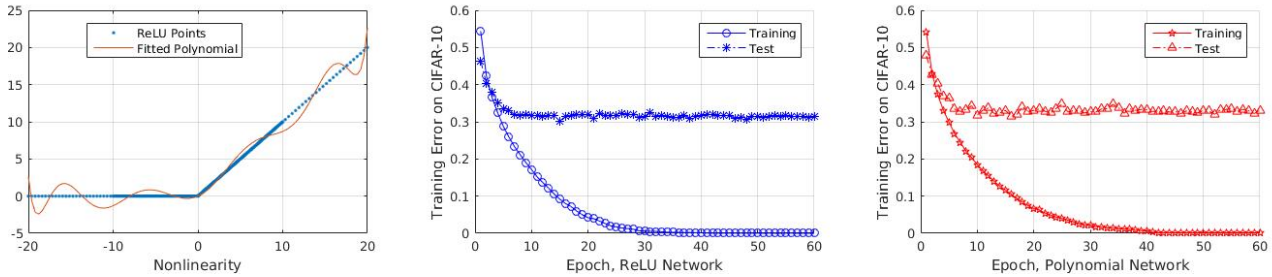


Figure 21: *A standard convolutional deep network is converted into a polynomial function by replacing each of the REL units with a univariate polynomial approximation of the RELU function. As long as the nonlinearity approximates ReLU well (especially near 0), the "polynomial network" performs similarly to a ReLU net. The polynomial shown in the inset on the left is of degree* 10.

simultaneously for the different tasks, forces the deep network to "find" *common constituent functions*. Multi-task learning has theoretical advantages that depends on compositionality: the sample complexity of the problem can be significantly lower (see [37]). The Maurer's approach is in fact to consider the overall function as the composition of a preprocessor function common to all task followed by a task-specific function. As a consequence, the *generalization error*, defined as the difference between expected and empirical error, averaged across the T tasks, is bounded with probability at least $1-\delta$ (in the case of finite hypothesis spaces) by

$$\frac{1}{\sqrt{2M}}\sqrt{ln|\mathcal{H}| + \frac{ln|\mathcal{G}| + ln(\frac{1}{\delta})}{T}}, \tag{61}$$

where M is the size of the training set, $\mathcal{H}$ is the hypothesis space of the common classifier and $\mathcal{G}$ is the hypothesis space of the system of constituent functions, common across tasks.

The improvement in generalization error because of the multitask 'structure can be in the order of the square root of the number of tasks (in the case of Imagenet with its 1000 object classes the generalization error may tyherefore decrease bt a factor ≈ 30). It is important to emphasize the dual advantage here of compositionality, which a) reduces generalization error by decreasing the complexity of the hypothesis space $\mathcal{G}$ of compositional functions relative the space of non-compositional functions *and* b) exploits the multi task structure, that replaces $ln|\mathcal{G}|$ with $\frac{ln|\mathcal{G}|}{T}$.

9.10 Polynomial approximation and deep polynomial networks

Nonlinear activations in a deep networks can be approximated up to an arbitrary accuracy over a bounded interval by a univariate polynomial of appropriate degree (see [2]). Since so much is known about polynomials several interesting results follow from this approximation properties such as a characterization of when deep convolutional networks can be exponential better than shallow networks in terms of representational power ([1]). However, not all properties of the polynomial approximants hold for the target function – the deep network. An example is the exact number of zeros of a square loss, which does not follow directly from approximation arguments.

We show in this section that the puzzling generalization properties of deep convolutional networks hold for networks with the same architecture in which all the RELU's have been replaced by a univariate polynomial approximation of degree 10. The resulting networks are obviously polynomial networks. Figure 21 shows very similar performance of polynomial and RELU networks with the same architecture in terms of training and testing error. Figures 21 and 23 show the same qualitative behavior of testing and training error on natural labeled data and on random data. Figure 24 shows the same puzzling lack of overfitting in the overparametrized case. Together Figures 22,23, 24 include all the puzzles discussed in [38]. The computational overhead posed by calculation of a polynomial rather than RELU is negligeable because computation of the activations is a very small fraction of the total computational budget involved in training.

The rest of the paper is dedicated to explaining this puzzling behavior of polynomial networks. Obviously the explanation for polynomial networks is likely to apply to standard deep networks.

9.10.1 Polynomial networks as linear mappings

The mapping from input to output of a deep polynomial network is

$$y = P_k(x) \qquad y \in \mathcal{R}^m, x \in \mathcal{R}^d, \tag{62}$$

where P_k is a polynomial in x of degree $k = 10 \times l$ – since 10 is the degree chosen for the univariate approximation of the RELU of Figure 21) and l is the number of layers. A generic polynomial $P_k(x)$ of degree k in d variables is in the linear space $\mathcal{H}_k$ of dimensionality $r = dim\mathcal{H}_k = \binom{d-1+k}{k}$: the latter is the number of monomials and thus number of coefficients of the polynomial. Such a generic polynomial can always be written (see Proposition 3.5 in [39]) as

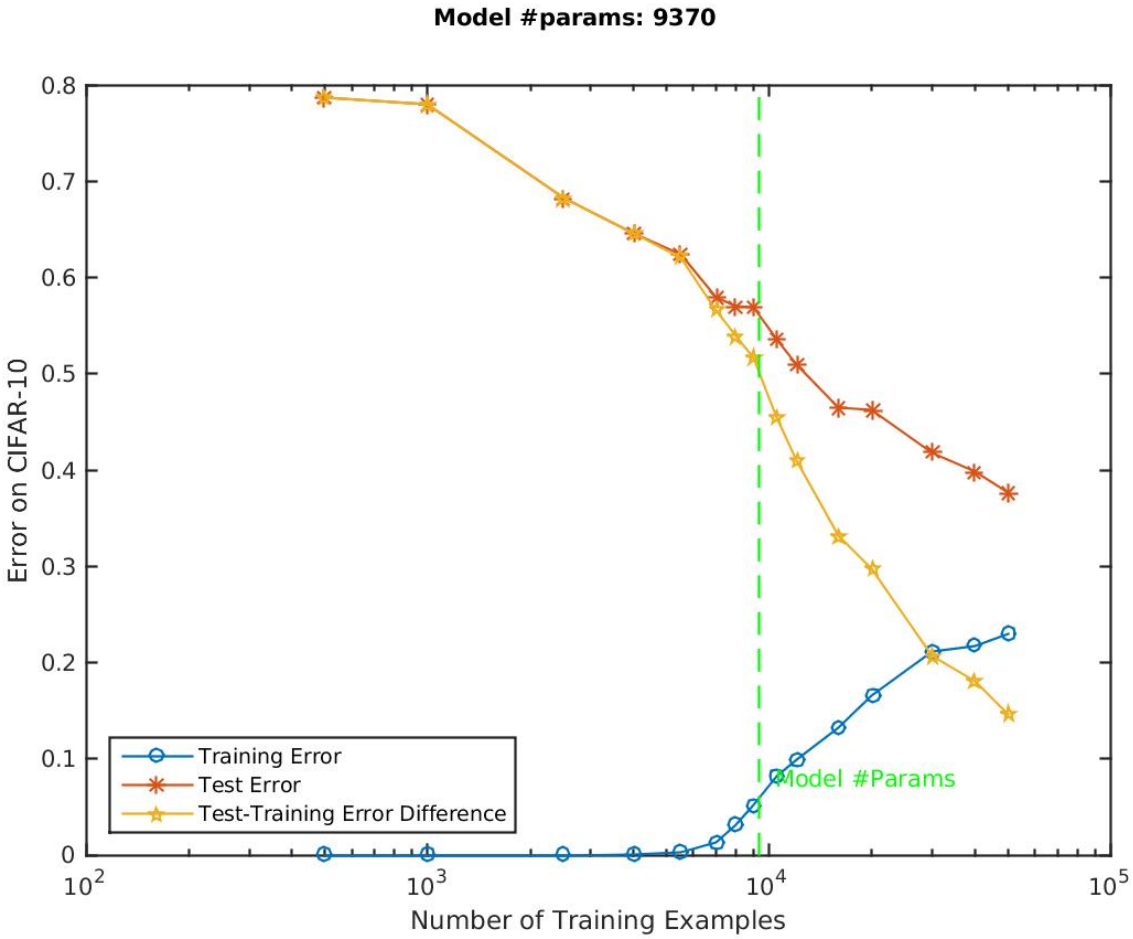


Figure 22: *The figure shows the behavior of a polynomial deep network trained on subsets of the CIFAR database. The network is a 5-layer all convolutional network (i.e., no pooling) with 16 channels per hidden layer , resulting in only* $W \approx 10000$ *weights instead of the typical* $300,000$*. Neither data augmentation nor regularization is performed.*

$$P_k(x) = \sum_{i=1}^{r} p_i(\langle w_i, x \rangle). \tag{63}$$

where p_i is a univariate polynomials of degree at most k and $\langle w_i, x \rangle)$ is a scalar product. Two remarks about Figure 25 are in order:

- in the case of hierarchically local functions and networks[1] – convolutional networks are a special case – the associated polynomials are very sparse with a number of monomials that grows linearly with d instead of exponentially ($\binom{d-1+k}{k} \approx k^d$).
- Notice that the effective r, is closely related to the *separation rank* of a tensor (see [1]).

Using the notation of [40], we can regard Equation 62 as a polynomial operator on the vector space $\mathcal{R}^d$. For instance, consider in Figure 25 the node in the second layer that receives x_1, x_2, x_3, x_4 inputs. The outputs of the node are of the form

$$\sum_{j=1}^{r'} p_j[W_1 \sum_{i=1}^{r} p_i(w_{1,i}x_1 + w_{2,i}x_2) + W_2 \sum_{i=1}^{r} p_i(w_{3,i}x_3 + w_{4,i}x_4)]. \tag{64}$$

with $p_h(x) = c_h p(x)$ and $p(x)$ is the polynomial activation function.

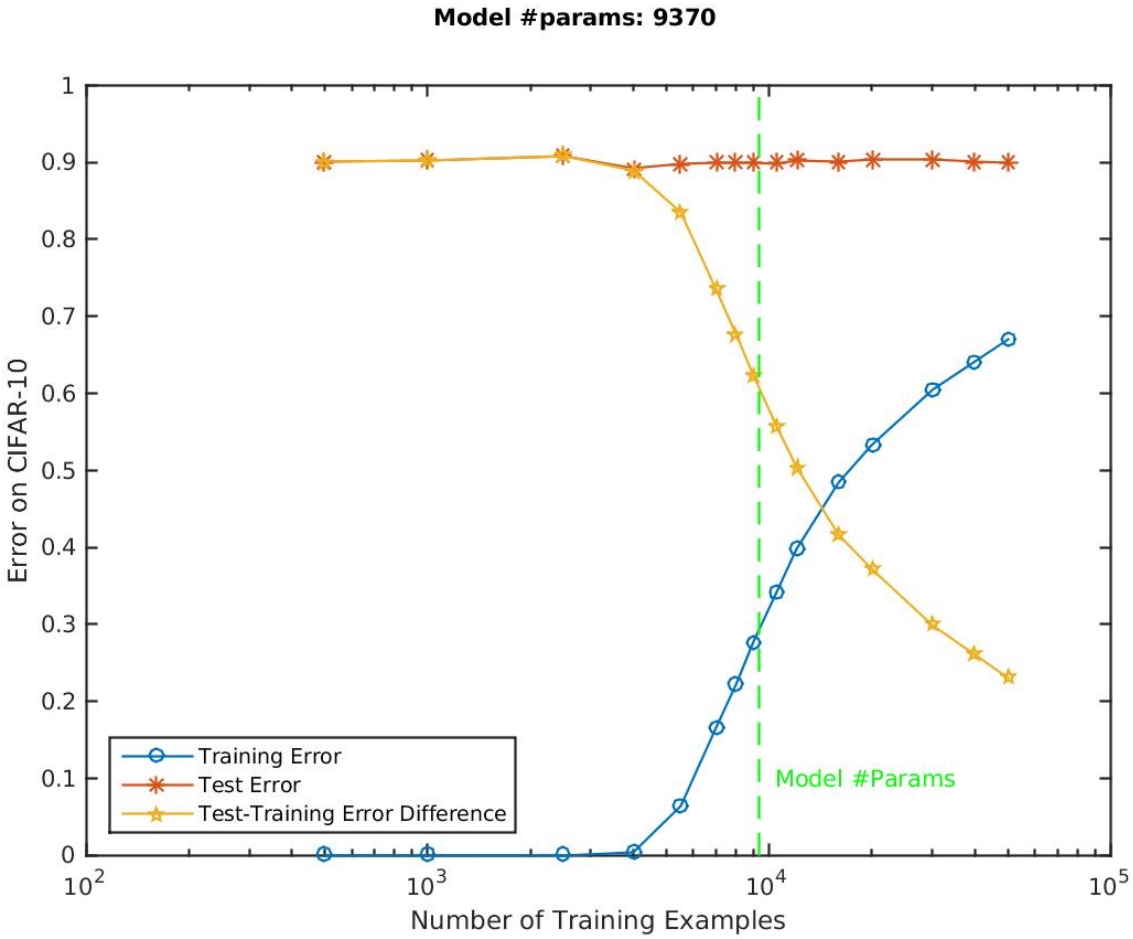


Figure 23: *The figure shows the behavior of a polynomial deep network trained on subsets of the CIFAR database in which the labels have been randomly scrambled. The network is a 5-layer all convolutional network (i.e., no pooling) with 16 channels per hidden layer, resulting in only* $W \approx 10000$ *weights instead of the typical* $300,000$*. Neither data augmentation nor regularization is performed.*

The key point in our argument is that a polynomial operator of degree k can be regarded as a linear mapping $\mathcal{L}$ from the (feature) space of all tensor products of x up to degree k to the space $\mathcal{Y}$.

Consider learning $\mathcal{L}$ from a set of example pairs x_i, y_i represented as columns of the matrices X, Y with X a d, n matrix and Y a m, n matrix. In this case $\mathcal{L}$ is a linear operator $\mathcal{L} : V^{\bigotimes k} \mapsto Y$ where $V^{\bigotimes k} = X \bigoplus X \bigotimes X \bigoplus \cdots X \bigotimes \cdots X = \cup_{i=1}^{k} V^i$. Notice that the dimensionality of each of the spaces of homogeneous polynomials V^i is high: $dim V^i = D = \binom{d+i}{d} \approx i^d$. Let us denote with L the m, D matrix corresponding to $\mathcal{L}$, with V the D, n matrix corresponding to example vectors x_i and with Y the m, n matrix of the y examples. The best solution in the L_2 sense of equation $LV = Y$ is then $L = YV^{\dagger}$. Since the matrix V is so large, only iterative techniques can be used to provide a solution. Two different constructions are sketched in the Appendix. The iterative techniques are closely related to the reparametrization of the coefficients of a polynomial in terms of weights of a multilayer linear network: each coefficient of a monomial (in the componets of x) is represented as product (with powers) of the weights. Several simple but relevant results follow from the setting of the learning problem as described in this section. We put all of them in a

Proposition 3. *Generalization in Deep Polynomial Networks*

- *In most of today's applications, data sets are much smaller than the number of coefficients of polynomials of degree* 10 *or larger in the input variables (because that number is exponential). Even convolutional networks implementing sparse polynomials will usually be overparametrized. Thus overparametrization is the standard situation.*
- *In the overparametrized case the optimal least square solution is the minimum norm pseudoinverse of the coefficients of the associated polynomial.*
- *Stochastic Gradient Descent as well as Gradient Descent converge to the minimum norm solution for appropriate initial condition in the usual parametriization of a polynomial (see Appendix).*

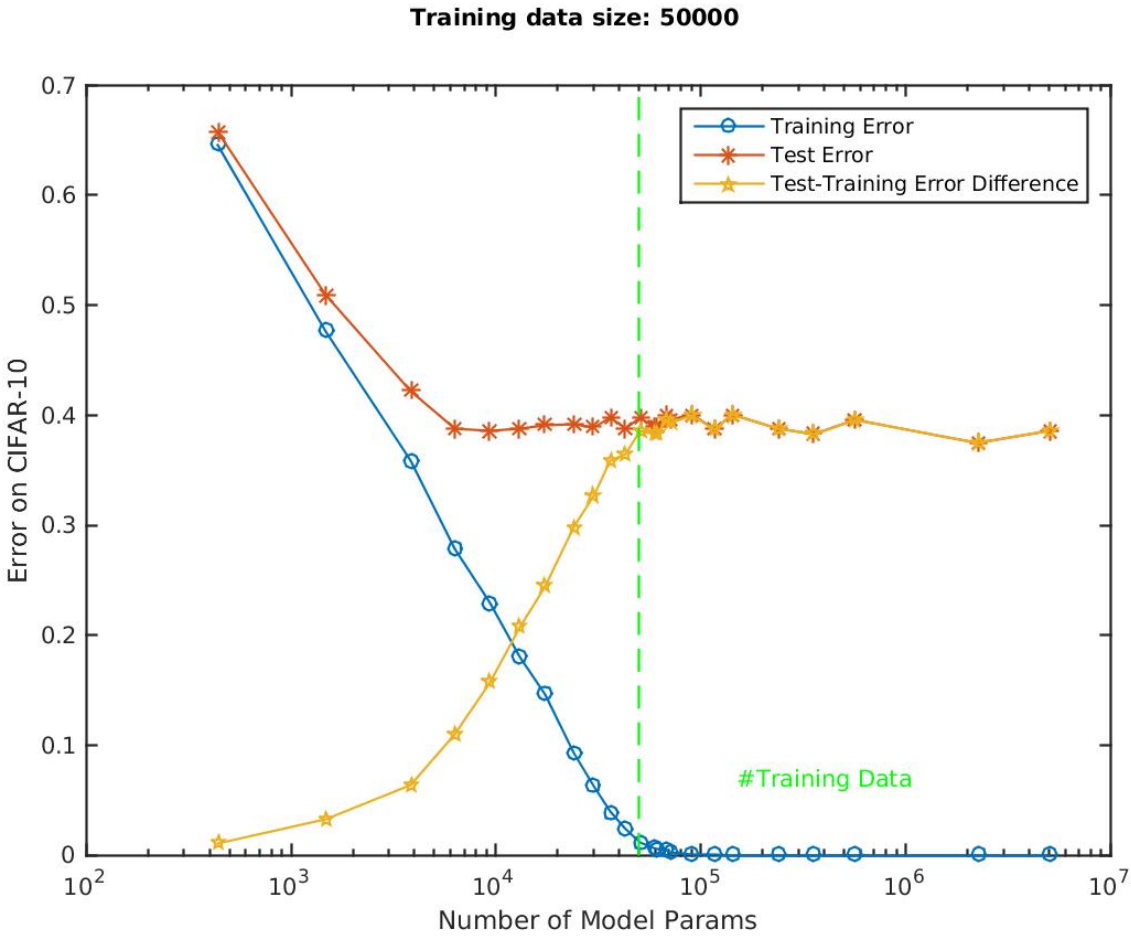


Figure 24: *The previous figures show dependence on N – number of training examples – for a fixed architecture with W parameters. This figure shows dependence on W for a fixed training set with N examples. The network is again a 5-layer all convolutional polynomial network. All hidden layers have the same number of channels. Neither data augmentation nor regularization is performed. The classical theory explains the generalization behavior on the left; the challenge is to explain the lack of overfitting for $W > n$. As shown here, there is zero error for $W \geq n$.*

- *Zero empirical error in the overparametrized case for natural and random label (Figure 23) follows from Bezout theorem[2] applied to the polynomial mapping $LV = Y$. Equivalently, it follows from the rank deficency of the system of linear equations $y_i = Lv_i$, assuming the system is consistent.*
- *The lack of overfitting for $W >> N$ in Figure 24 follows from properties of the pseudoinverse solution $L = YV^{\dagger}$.*
- *Improving test error for larger and large n in Figure 22 follows from the solution L of the linear problem being optimal in L_2.*
- *From older results on linear multilayer networks, we know that*
  - *optimization over weights and optimization over coefficients of the polynomial provide the same result (see [41])*
  - *the dynamics is different for direct optimization of the coefficients of the polynomial vs optimization wrt weights: in the latter we expect from results on linear networks[42] a learning dynamics with stage-like transitions between longer stable periods, that depends on the singular values associated with the input-output correlations and the input autocorrelations, where the input is the space V of tensor products of x.*
  - *there is a set of global degenerate minima wrt coefficients of the polynomial if the linear system of equations $y_i = Lv_i$ is not inconsistent ([41, 2, 43]). Depending on the initial conditions a minimum norm solution by GD or SGD is likely (K. Kawaguchi, personal communicatio) is likely*
  - *For linear ResNets (the most interesting to us) Theorem 2.1 of [44] states that the optimal solution has small norm (defined as the maximum – over the network layers – of the Frobenius norm of the matrix of weights for each layer) is small. Furthermore this norm is inversely proportional to the number of layers.*

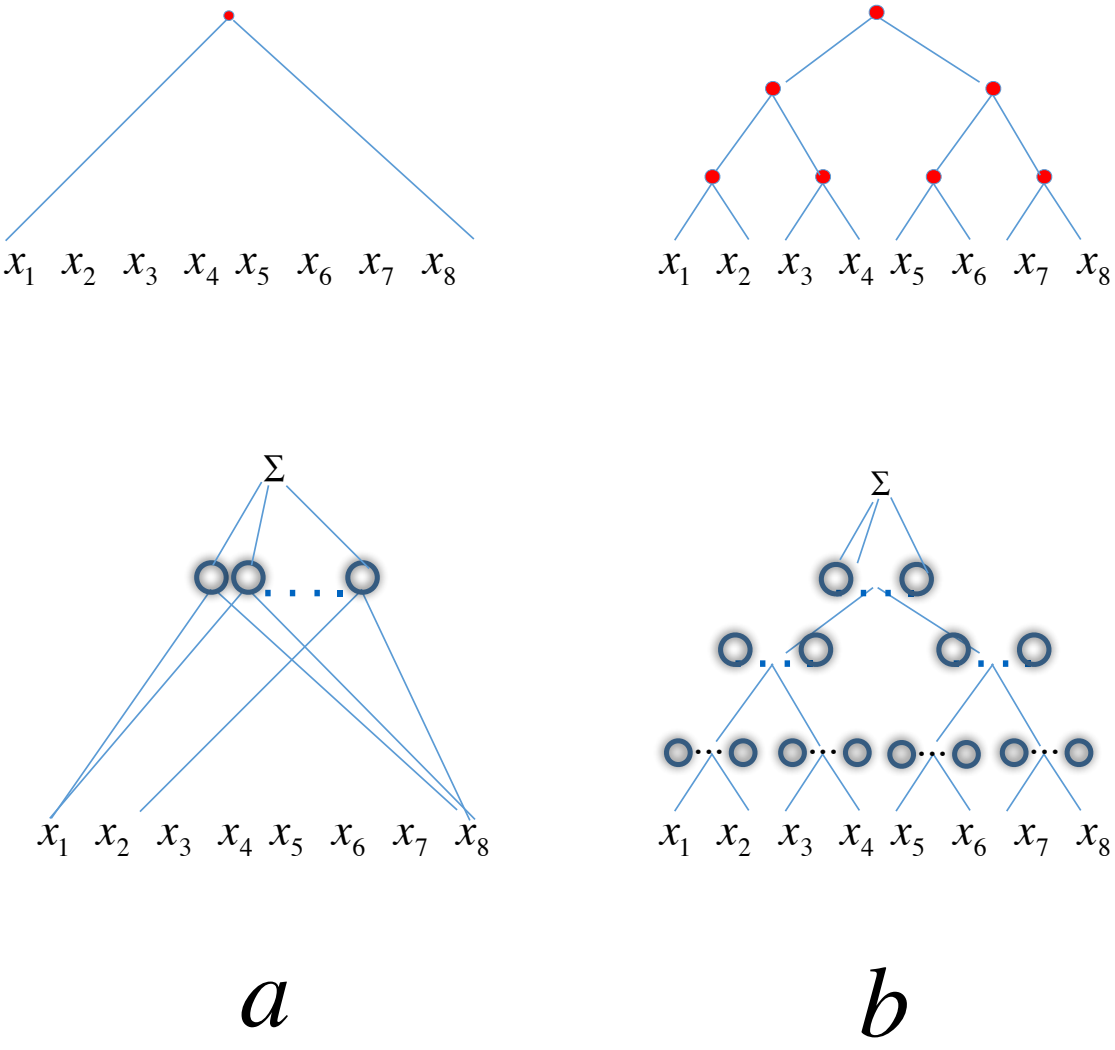


Figure 25: *The top graphs are associated to functions; each of the bottom diagrams depicts a polynomial network approximating the function above. In a) a shallow polynomial in 8 variables and N units approximates a function of 8 variables $f(x_1, \cdots, x_8)$. If the activation for each of the hidden units is a polynomial of degree 10 then $N = \binom{8+10}{8}$ since a polynomial in d variables of degree k can be written as a linear combination of the same number N of univariate polynomials of degree k in $(\langle w, x \rangle + b)$. The bottom of inset b) shows a binary tree hierarchical network in $n = 8$ variables, which approximates well functions of the form $f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$ as represented by the binary graph above. In the approximating network each of the $n - 1$ nodes in the graph of the function corresponds to a set of polynomial units. In a binary tree with d inputs, there are $log_2 d$ levels and a total of $d - 1$ nodes. The locality of the constituent functions – that is their low dimensionality – means that the overall polynomial computed by the network is sparse.*