Does SGD Implicitly Optimize for Smoothness?

Václav Volhejn, Christoph Lampert

IST Austria

Abstract. Modern neural networks can easily fit their training set perfectly. Surprisingly, despite being "overfit" in this way, they tend to generalize well to future data, thereby defying the classic bias–variance trade-off of machine learning theory. Of the many possible explanations, a prevalent one is that training by stochastic gradient descent (SGD) imposes an implicit bias that leads it to learn simple functions, and these simple functions generalize well. However, the specifics of this implicit bias are not well understood.

In this work, we explore the *smoothness conjecture* which states that SGD is implicitly biased towards learning functions that are smooth. We propose several measures to formalize the intuitive notion of smoothness, and we conduct experiments to determine whether SGD indeed implicitly optimizes for these measures. Our findings rule out the possibility that smoothness measures based on first-order derivatives are being implicitly enforced. They are supportive, though, of the smoothness conjecture for measures based on second-order derivatives.

1 Introduction

Classical machine learning wisdom suggests that the expressive power of a model class (its *capacity*) should be carefully balanced with the amount of available training data: if the capacity is too low, learned models will *underfit* and not manage to fit the training set, let alone the test set. If the capacity is too high, learned models do fit the training set, but they *overfit* to spurious patterns and fail to represent the underlying trend, again failing to generalize well to the test set. Thus, the learned models generalize best when the capacity is in a sweet-spot somewhere between underfitting and overfitting. This observation is also known as *bias–variance trade-off*.

Several researchers have observed that neural networks seem to defy the bias–variance trade-off: increasing model capacity often improves generalization performance, even if the network is already apparently "overfit". This phenomenon had first been reported more than 20 years ago, e.g. [15,5], but it has only begun receiving wider attention in recent years. This started with the work of [22], who showed that plotting the test loss as a function of model capacity (represented by the hidden layer size) does not yield the U-shaped curve predicted by the bias–variance trade-off, but starts to decrease again for high model class capacities. The authors then conjecture that the surprising generalization performance of "overfit" neural networks might be due to implicit regularization in the training

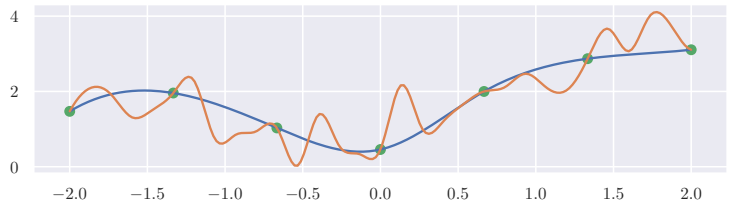


Fig. 1: A non-smooth function (orange) and smooth function (blue) interpolating a one-dimensional dataset.

process: while the training objective only penalizes the prediction quality, the optimization process nevertheless prefers solutions that have "small complexity" and therefore generalize well.

It is still an open question, though, what exactly the implicitly regularized complexity measure is. In this work, we explore the conjecture (put forward, e.g., in [19]) that it is "smoothness" which is implicitly regularized: stochastic gradient descent tends to produce functions that are not needlessly "rough" or "bumpy". For an illustration, see Figure 1. While this *smoothness conjecture* is intuitively appealing, it is not clear so far how the intuitive concept of "smoothness" would be correctly formalized mathematically. This is especially a problem because in high dimensions, as common in machine learning, there are many possible notions of smoothness for a function.

Our goal in this work is to make progress towards a formal analysis of the smoothness conjecture. Specifically, our main steps are the following:

- We define four measures that express "smoothness" of a trained neural network; two rely on first-order information, two on second-order information.
- We introduce two experimental settings that allow us to assess compatible the smoothness conjecture for each of these measures is with empirical observations.
- Based on our experimental results, we argue that first-order smoothness measures can be excluded as candidates for SGD's implicit regularization, whereas second-order methods are promising candidates.

2 Related work

To the best of our knowledge, the first modern paper that observed the unexpected generalization behavior of neural networks is [22], where the authors focus on the fact that the test loss keeps decreasing with the number of hidden units. This was later followed by more refined analyses (e.g. [2,4,27]), which observed a *"double descent"* behavior: for low-capacity model classes, the standard reasoning of over- and underfitting holds. For model classes of very high capacity, though, where the training error can be reduced to zero, i.e. the data is interpolated, higher model class capacity corresponds to further reductions of the test loss. Later work [21] confirmed the findings in extensive experiments

and observed that a double descent occurs not only as a function of model size but also the number of training epochs.

In [28] it was shown that modern deep convolutional neural networks are able to fit datasets even with random labels, and it is thereby easy to find models of small training error that do not generalize well. Consequently, the explanation for the unexpectedly good generalization behavior cannot be that all interpolating models generalize equally well. Instead, it must be the specific solutions found by standard network training (using stochastic gradient descent) that have these favorable properties.

A popular form of explanation introduced already by [22] is that the training procedure is implicitly biased towards solutions with low complexity. Subsequently, most works concentrated on the question which property of trained neural networks it could be that would make them generalize well. Suggestions include the *sharpness* of the reached loss function minimum [13], *distance from initialization* [20], *Fisher–Rao norm* [18], as well as various measures based on parameter norms [3,23]). However, an extensive empirical comparison in [10] showed that many of the proposed measures are not positively correlated with generalization quality. Therefore, the question of how to *enforce* generalization for high-complexity model classes remains so far unsolved.

Some of the smoothness measures that we discuss later have been studied previously in other contexts. We postpone the discussion of this related work to Section 3.4, after we have presented the measures in technical form.

In this work, we do not try to solve the question of which complexity measure should best be minimized for neural networks to generalize well, but the question which such measure SGD actually implicitly regularizes, if any. Our approach is inspired by [19], who observe that under certain conditions, training shallow ReLU networks in the one-dimensional setting using gradient descent yields "simple" functions that are close to a piecewise-linear interpolation of the training data. The author do not explore analogs for real networks with high-dimensional inputs, though. Another work that is related to our analysis is the recent preprint [14], where also the smoothness of trained neural networks is studied. The authors find that overparametrized networks interpolate almost linearly between the samples, which is consistent with our findings. The work does not answer the question if smoothness is actively minimized by SGD, though.

3 Does SGD implicitly optimize for smoothness?

We study the implicit regularization properties of stochastic gradient descent training for neural networks in a standard setup of supervised learning. We adopt a regression setting with input set $\mathbb{R}^d$. and output set $\mathbb{R}$. Assuming a fixed but unknown data distribution $\mathbb{P}$, the goal is to learn a function $f\colon \mathbb{R}^d \to \mathbb{R}$ with low *expected loss*,

$$\mathcal{L}(f) = \mathbb{E}_{(x,y)\sim\mathbb{P}}(f(x) - y)^2. \tag{1}$$

While the data distribution is unknown, we do have access to a training set $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ consisting of independent and identically distributed

(i.i.d.) samples from $\mathbb{P}$. This allows us to define the *training loss*

$$L(f) = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2. \tag{2}$$

For a fixed model class, $\mathcal{F}$, e.g. the set of ReLU networks of a fixed architecture, we want to select (*learn*) a function $f \in \mathcal{F}$ which minimizes the training loss. We are primarily interested in models that perfectly fit or *interpolate* the training data, meaning the learned function satisfies $L(f) = 0$. For numerical reasons, we only require $L(f) < \varepsilon$ with a small ε (e.g. 10^{-5}) in practice. Typically, if the model class is rich enough to contain any model that fulfills this condition, then it contains many of them. The informal *smoothness conjecture* is:

> *When the model class is a set of neural networks that is rich enough to interpolate the training data and we use stochastic gradient descent for training, then the resulting trained model is not an arbitrary minimizer of the training loss, but among the smoothest possible ones.*

In this work, we aim towards a better understanding of the validity of this conjecture. First, we formalize several smoothness measures, which makes it possible to treat the above conjecture as a mathematical rather than just an informal statement. Then, we provide experimental evidence that support the smoothness conjecture for some smoothness measures while refuting it for others.

3.1 Measuring the smoothness of a function

In machine learning, the notion of smoothness of a function is often used intuitively (e.g. [4,11,14]) and it is rarely defined formally. In this section we formulate four measures that assign scalar smoothness values to functions $f\colon \mathbb{R}^d \to \mathbb{R}$. To be precise, the measures we define quantify *roughness* or the *absence of smoothness*, as we will use the convention that small values (close to 0) indicate smooth functions, whereas large values indicate functions with little smoothness. This convention is, unfortunately, necessary to be compatible with most of the prior literature. The non-negativity reflects the qualitative use of the term smoothness as a single-sided bounded measure: there is a limit on how smooth a functions can be (e.g. attained by constant functions), but there is no obvious limit to how non-smooth it could be.

The measures we discuss can be classified into two categories: *first-order* and *second-order* smoothness measures. First-order measures are based on properties of the first-order derivatives (gradients) or differences between function values of f. Second-order measures are based on second-order derivatives, or differences between gradients of f.

3.2 First-order smoothness measures

Inspired by the common procedure for linear models, a simple way to formalize smoothness is to identify it with *steepness*: if a function is very steep (has a

large gradient magnitude), then it is not very smooth. For non-linear functions, which we are interested in, the gradient varies for different input arguments. To obtain a scalar measure, we take the expected value of the Euclidean norm of the function's gradient with respect to the underlying data distribution.

Definition 1 (Gradient norm). *Let $\mathbb{P}_X$ be a probability distribution over $\mathbb{R}^d$ and let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a function that is differentiable almost everywhere with respect to $\mathbb{P}_X$. We define the gradient norm of f with respect to $\mathbb{P}_X$ as*

$$\mathrm{GN}(f) = \mathbb{E}_{x\sim\mathbb{P}_X} \|\nabla_x f(x)\|. \tag{3}$$

GN is non-negative and it is 0 for those functions whose gradient is zero almost everywhere. This, in particular, includes constant functions, but also piece-wise constant functions as long as the set where f changes values has measure zero according to $\mathbb{P}_X$.

Because it is defined as an expected value over the data distribution, we can approximate $\mathrm{GN}(f)$ by random sampling: let $x_1, \ldots, x_N$ be N data samples that were not used for training f, then we set

$$\widehat{\mathrm{GN}}(f) = \frac{1}{N}\sum_{i=1}^{N} \|\nabla_x f(x_i)\|. \tag{4}$$

For our experiments we use $N = 1000$ and we use automatic differentiation to compute the gradient.

An alternative approach for characterizing smoothness that avoids the condition of differentiability is to study changes of the function values along one-dimensional line segments. For this, we define

Definition 2. *Let $f\colon \mathbb{R}^d \to \mathbb{R}^k$ be a (potentially vector-valued) function and let $a, b \in \mathbb{R}^d$. We define a* line segment of f from a to b *to be*

$$f_{[a,b]}(t) = f((1-t)a + tb). \tag{5}$$

Studying the curve induced by the function values on any such line segment, we obtain an intuitive measure of smoothness. If the curve is straight and short, the underlying function is smoother than if the curve is wrinkled and long. Mathematically, we define the *function path length* as the expected value of the *total variation* over all line segments of f with end points distributed according to the data distribution:

Definition 3 (Function path length). *Let $\mathbb{P}_X$ be a probability distribution over $\mathbb{R}^d$ and let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a function. We define the* function path length *of f with respect to $\mathbb{P}_X$ as*

$$\mathrm{FPL}(f) = \mathbb{E}_{a,b\sim\mathbb{P}_X} \mathrm{TV}(f_{[a,b]}) \tag{6}$$

where the total variation of a function $g : [0,1] \to \mathbb{R}^k$ is defined as

$$\mathrm{TV}(g) = \sup_{P\in\mathcal{P}} \sum_{i=1}^{|P|} \|g(t_i) - g(t_{i-1})\| \tag{7}$$

with $\mathcal{P}$ denoting the set of all partitions of the interval $[0,1]$:

$$\mathcal{P} = \Big\{ P = (t_0, t_1, \ldots, t_{|P|}) \,\Big|\, 0 = t_0 < t_1 \cdots < t_{|P|} = 1 \Big\}. \tag{8}$$

FPL is non-negative by construction and minimized (with value 0) by all constant functions.

As GN before, the fact that FPL is defined in terms of an expectation operation over the data distribution makes it possible to derive a sampling-based approximation. Let $(a_i, b_i)_{i=1,\ldots,N}$ be N pairs of data points that were not used during the training of f. Then we set

$$\widehat{\mathrm{FPL}}(f) = \frac{1}{N} \sum_{i=1}^{N} \widehat{\mathrm{TV}}(f_{[a_i,b_i]}) \tag{9}$$

where $\widehat{\mathrm{TV}}$ approximates TV using a regular subdivision of the input interval:

$$\widehat{\mathrm{TV}}(f_{[a,b]}) = \sum_{i=1}^{n-1} |f(t_i) - f(t_{i-1})| \tag{10}$$

with $t_i = \frac{i}{n-1}a + \big(1 - \frac{i}{n-1}\big)b$ for $i \in \{0, \ldots, n-1\}$. For our experiments, we use $N = 1000$ and $n = 100$.

While first-order smoothness measures are intuitive and efficient, they also have some shortcomings. In particular, neither the gradient norm nor the function path length can distinguish between some functions which we would not consider equally smooth. For example, take $f(x) = x$ on $[0,1]$ and $g(x) = 0$ on $[0, \frac{1}{2}]$ and $g(x) = 2x - 1$ on $[\frac{1}{2}, 1]$ under a uniform data distribution. Both functions have identical function path length and gradient norm, even though intuitively one would consider f smoother than g. This problem can be overcome by looking at measures that take second-order information (i.e. curvature) into account.

3.3 Second-order smoothness measures

A canonical choice for a second-order smoothness measure would be to compute properties (e.g. the Frobenius norm or operator norm) of the Hessian matrix. Unfortunately, this is not tractable in practice, because of the high computational effort of computing the Hessian matrix many times, as well as the memory requirements, which are quadratic in the number of input dimensions.

Instead, the first measure we propose relies on an analog of the construction used for the function path length, now applied to the function's gradient instead of its values.

Definition 4 (Gradient path length). *Let $\mathbb{P}_X$ be a probability distribution over $\mathbb{R}^d$ and let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. We define the gradient path length as*

$$\mathrm{GPL}(f) = \mathbb{E}_{\,a,b\sim\mathbb{P}_X}\, \mathrm{TV}((\nabla_x f)_{[a,b]}) \tag{11}$$

GPL is non-negative by construction. It vanishes on constant functions, but also on linear (more precisely: affine) ones. To approximate GPL in practice, we use the same construction as for FPL, where the occurring gradients are computed using automatic differentiation.

A special situation emerges for two-layer ReLU networks, i.e. functions of the form

$$f(x) = \langle w^{(2)}, a\rangle + b^{(2)} \text{ with } a_i = \mathrm{ReLU}\big[\langle w_i^{(1)}, x\rangle + b_i^{(1)}\big] \quad \text{for } i = 1, \dots, h, \tag{12}$$

where h is the number of hidden units in the network and $w_1^{(1)}, \dots, w_h^{(1)}$ and $w^{(2)}$ are weight vectors of suitable dimensions and $b_1^{(1)}, \dots, b_h^{(1)}$ and $b^{(2)}$ are scalar bias terms. For these, we can compute a measure of second-order smoothness explicitly from the parameter values.

Definition 5 (Weights product). *Let $f_\theta \colon \mathbb{R}^d \to \mathbb{R}$ be a two-layer ReLU network with parameters $\theta = (W^{(1)}, b^{(1)}, w^{(2)}, b^{(2)})$, where $W^{(1)} = (w_1^{(1)}, \dots, w_h^{(1)})$ with $w_i^{(1)} \in \mathbb{R}^d$ and $b^{(1)} = (b_1^{(1)}, \dots, b_h^{(1)})$ with $b_i^{(1)} \in \mathbb{R}$ for $i = 1, \dots, h$, as well as $w^{(2)} \in \mathbb{R}^h$ and $b^{(2)} \in \mathbb{R}$. We define the* weights product *measure as*

$$\mathrm{WP}(f_\theta) = \sum_{i=1}^{h} |w_i^{(2)}| \cdot \big\|w_i^{(1)}\big\| \tag{13}$$

where $w_i^{(2)}$ indicates the i-th entry of the vector $w^{(2)}$ for any $i = 1, \dots, h$.

WP is non-negative by construction, and takes the value 0 on networks where for each neuron in the hidden layer either all incoming weights or the outgoing weight are zero, with arbitrary values of the bias terms. From Equation (12) one sees that all constant functions can be expressed this way.

A small computation establishes that for one-dimensional inputs, WP is equal to the total variation of the derivative, under the assumption that the positions at which the hidden units switch between deactivation and activation $(-b_i^{(1)}/w_i^{(1)})$ are unique. In higher dimensions, each summand in (13) is still the norm of the difference of the gradients on the two sides of the ReLU activation function. Thus WP is a second-order measure, based on the changes of the gradient.

In contrast to the previous smoothness measures, WP is only defined for two-layer ReLU networks and does not take the underlying data distribution into account. Its advantage, though, is that it can easily be computed exactly, without having to rely on sampling-based approximations as for the other measures.

3.4 Smoothness measures in related work

The measure we call *gradient norm*, as well as minor variants, were explored in multiple prior works, e.g. [6,24,25,26]. Generally, the findings are that a small average norm of the Jacobian, i.e. the gradient in the scalar setting, can lead to improved generalization. In [17], a measure of "rugosity" (roughness) is proposed based on the learned function's Hessian matrix. The authors also discuss a Monte Carlo approximation of this quantity, which resembles our notion of *gradient path*

length, with the main difference being that it uses local perturbations instead of a line segment. Even closer is the smoothness measure in [14] which also measures how the gradient of a function changes when interpolating between two samples.

The *weight product* measure is an analog of the *path-regularizer* of [23] for two-layer ReLU networks. In that work, the measure is proposed for training-time regularization, not as a post-hoc smoothness measure. To our knowledge, the *function path length* measure has not been used in the context of neural networks, but a similar constructions was suggested, e.g., for audio signals [9].

4 Experiments

We report on our experiments that shed light on the validity of the smoothness conjecture in general, and with respect to the four proposed smoothness measures in particular. Note that the naive approach of simply checking the numeric values of the smoothness measures is not possible, because we do not know what reference value to compare them to. Ideally, this would be the smallest achievable smoothness value for any network of the studied class on the provided data. Unfortunately, we cannot easily compute these on high-dimensional data, only derive some lower bound (see Table 2).

Instead, we use two proxy setups that we consider contributions of potentially independent interest, as they would also be applicable to other measures besides smoothness. First, we study how monotonically the measures behave when networks are trained with increasing amounts of data. If the smoothness conjecture is fulfilled, one would expect perfect monotonicity, see the discussion in Section 4.2. Second, we analyze whether substantially smoother models exist than the one produced by SGD that nevertheless interpolate the data. Under the smoothness conjecture, this should not be the case, see Section 4.3.

Before reporting on the results of the experiments, though, we introduce the experimental setup.

4.1 Experimental setup

The surprising generalization abilities have been observed for networks of all sizes. We restrict our own analysis to small networks, because the more efficient experiments allows us to try more different settings and perform multiple reruns to gain statistical power. Specifically, we use fully connected ReLU networks with one hidden layer of size $h = 256$. We train the networks using mini-batch stochastic gradient descent with a batch size of 64, and, unless specified otherwise, a learning rate of 0.01. For network initialization, the network's bias terms are initially set to 0. The weights in each of the two layers are initialized by drawing uniformly from the interval $[-\ell, \ell]$ with $\ell = \sqrt{\frac{6\alpha}{n_{in}+n_{out}}}$. n_{in} is the number of input units of the layer, n_{out} is the number of output units, and α is an *initialization scale*. For $\alpha = 1$, this initializer would reduce to the widely used Glorot uniform initializer [7]. We use $\alpha = 0.01$ instead, as it has been observed

in [19] that a smaller initialization scale generally leads to smoother learned functions, and this is also consistent with our own findings.

Many existing works use the number of training epochs as stopping criterion. This is not ideal for our setting, as we observed that training models of different complexity, e.g. with different regularization terms, for the same number of epochs leads to large differences in the achieved training losses and how close to convergence the models actually are. Instead, we use a threshold of 10^{-5} on the training loss as the stopping criterion. This choice ensures that the models fit the training set almost perfectly and have converged to a comparable level.

All experiments were performed using the TensorFlow framework [1], assisted by the *Sacred* package [8] for enhanced reproducibility. Our code will be publicly available.

As a data source, we use the MNIST dataset of handwritten digits [16] in the following way. For any pair of digits (a, b) with $0 \leq a < b \leq 9$, we construct a training set consisting of 5,000 images of digit a with target value -1 and 5,000 images of digit b with target value 1. This results in $\binom{10}{2} = 45$ regression problems, which we call the *MNIST-binary* problem set. By solving multiple small regression problems instead of a single large one, we hope to reduce variance and gain more confidence that the observed trends are not just due to randomness. As data for computing the smoothness measures, we use subsets of the MNIST test set with the corresponding digits.

4.2 Monotonicity

As a first test of the hypothesis that smoothness is implicitly enforced during neural network training, we use the following observation. Imagine two training sets, D and D', where D' is identical to D, except that some more data points have been added to it. Denote by f and f' the smoothest possible functions in a hypothesis set that interpolate the data in D and D', respectively. Then f' cannot be smoother than f, because adding training samples means adding constraints to the interpolation problem, and the minimizer over a more constrained set can only achieve a higher or equal objective value.

Our experiments verify empirically if this phenomenon indeed occurs for different model classes and the smoothness measures of Section 3.1. We first select a dataset $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ that all studied model classes are able to interpolate. Then we construct an increasing sequence of datasets $D_1, \ldots, D_n$ with $D_i \subsetneq D_{i+1}$ for $i = 1, \ldots, n-1$, and train models on each of these datasets, obtaining functions $f_1, \ldots, f_n$. For any smoothness measure S (with the convention that a lower value of S means a smoother function), we expect to obtain

$$S(f_1) \leq S(f_2) \leq \cdots \leq S(f_n) \tag{14}$$

if the smoothness conjecture holds for S. As a quantitative measure of how close we are to all inequalities holding, we use the Kendall rank correlation coefficient, $\tau \in [-1, 1]$, which reflects the number of inversions in the sequence, see [12]. A value of $\tau = 1$ means perfect accordance with (14).

Table 1: Monotonicity score (Kendall's τ) for the 45 MNIST-binary datasets. Standard deviation is not listed for WP and GPL because these measures reach the maximum value of τ for every dataset.

Smoothness measure	GN	FPL	GPL	WP
Kendall's τ	0.16 ± 0.32	0.07 ± 0.46	1.0	1.0

For each of the 45 MNIST-binary tasks, we use training set sizes $N = \{64, 128, 256, 512, 1024, 2048, 4096, 8192\}$. To lower the variance, we repeat the experiment three times for each dataset. Therefore, we obtain a total of $3 * 45$ values of τ.

Table 1 summarizes the results as mean and standard deviation over the obtained τ values. We see that for the first-order smoothness measures, GN and FPL, the change in function smoothness is highly fluctuating and only weakly correlated with growing dataset size. In contrast, the rank correlation is consistently at its maximum value for the second-order measures, GPL and WP.

4.3 Optimality

In this section, we take a second look at the question of whether smoothness is implicitly optimized by SGD training and if yes, which notion of smoothness exactly that is. For this, we take an exclusion approach: we can be sure that a complexity measure S is *not* being regularized implicitly during training, if we are able to find another model that performs equally well on the training set but is substantially smoother according to this measure than that found by SGD.

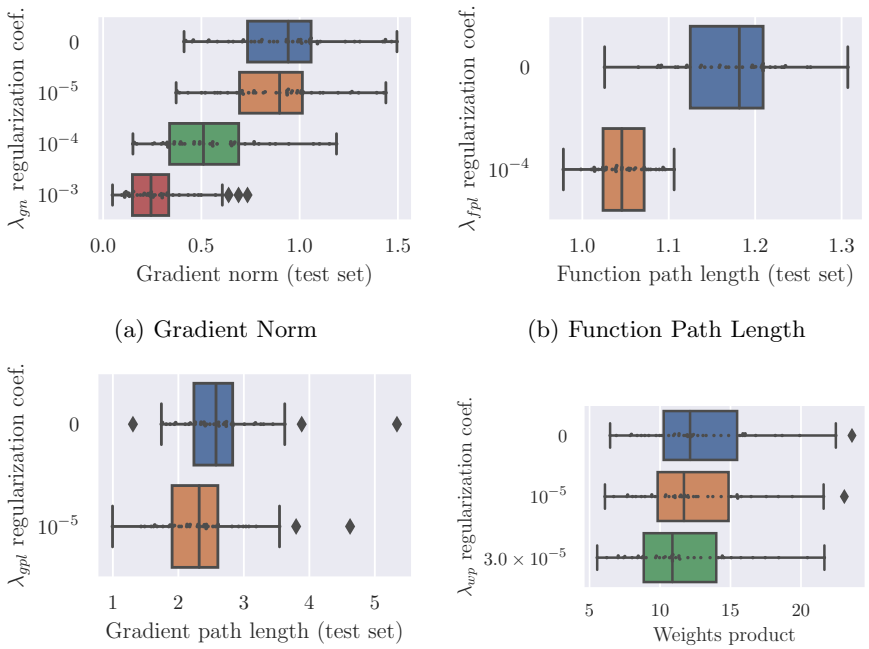
To search for such smoother models, we rely on *explicit* regularization. During network training, we replace the original loss function L with a regularized version L_{reg}, in which we penalize high values of S:

$$L_{reg}(f) = L(f) + \lambda S(f) \tag{15}$$

where $\lambda > 0$ is a regularization coefficient.

When S is expensive to compute, network training is slowed down considerably, as the measure and its gradient have to be evaluated in every training step. We therefore use stochastic variants of the smoothness measures that always use the current training batch as data points. Note that to evaluate the trained model's smoothness, we still use the full variants on test data. Furthermore, we use a learning rate of 0.1 for the two path length measures instead of the default value of 0.01. This is merely for practical reasons as it ensures training converges in a reasonable time; we did not observe any negative side effects of this change.

Figure 2 shows the results of all experiments as box-whisker plots for the 45 MNIST-binary datasets. The first row in each plot ($\lambda = 0$) corresponds to training the unregularized objective, i.e. plain SGD training. The other rows reflect training with different amounts of regularization strength. Setups in which the explicit regularization was too strong to reach the interpolation regime (i.e. training error remained above 10^{-5}) are not reported and not included in the analysis.



(a) Gradient Norm

(b) Function Path Length

(c) Gradient Path Length

(d) Weights Product

Fig. 2: Effect of explicit smoothness regularization for training on MNIST data for different smoothness measures (subfigures) and regularization strengths (y-axis). Lower values indicate smoother models.

Table 2 contains a numeric summary of these results. The columns *Unregularized mean* and *Regularized mean* show the mean value of the respective smoothness measure across the 45 MNIST-binary tasks (the full score distribution was already provided in Figure 2). The regularized mean is computed only from the models with the largest reported regularization coefficient. *Lower bound*, l_S, is a bound on the smallest value that the corresponding measure, S, can take on an interpolating model from the studied model class. For GN, GPL and WP, only the trivial bound 0 is readily available. For FPL, we know that data pairs of identical output value contribute 0 to Equation (9), while data pairs of opposite output values contribute at least 2, so a lower bound on FPL for balanced data is 1. The normalized ratio is computed as

$$r_{norm} = \frac{1}{45} \sum_{i=1}^{45} \frac{S(f_i^{reg}) - l_S}{S(f_i^{unreg}) - l_S}, \tag{16}$$

where f_i^{reg} and f_i^{unreg} are the results of training models on the i-th task with and without regularization, respectively.

The plots and table show a clear trend: for the first-order smoothness measures, adding explicit regularization to the training objective results in models that have equally small training loss yet much higher smoothness (GN and FPL

Table 2: Numeric summary of explicit regularization experiment results. For each measure, we report the results for the highest regularization coefficient for which we were still able to train the models to achieve 10^{-5} training loss. For an explanation of the rows, see the main text.

Smoothness measure	GN	FPL	GPL	WP
Unregularized mean μ_{unreg}	0.93	1.17	2.62	12.84
Regularized mean μ_{reg}	0.29	1.05	2.35	11.57
Lower bound l_S	0	1	0	0
Normalized ratio r_{norm}	0.31 ± 0.13	0.33 ± 0.27	0.89 ± 0.06	0.90 ± 0.03

are reduced by approximately 70%). Consequently, we can reject the conjecture that SGD implicitly optimizes for these measures. Note that this finding is not incompatible with results in the literature that enforcing a small norm of the gradient can positively impact generalization [24,25,26], as that is just a sufficient criterion, not a necessary one.

For the second-order smoothness measures the results show the opposite effect. By including explicit regularization, we were not able to substantially increase the models' smoothness (GPL and WP are reduced by approximately 10%). Formally, our result cannot be taken as proof that no substantially smoother models exist. After all, we might just not have been able to find them using the explicit regularization procedure. Nevertheless, the results do support the conjecture that SGD does have a regularizing effect on neural network training, and they concretize the formulation of the smoothness conjecture: the enforced smoothness is likely of second-order type.

5 Conclusion

In this work, we empirically studied the conjecture that training neural networks by stochastic gradient descent results in models that do not only have a small training loss, but that at the same time are very smooth, even when the training objective does not explicitly enforce the latter property. If correct, the conjecture would be a major milestone towards a better understanding of the generalization properties of neural networks.

After introducing four different smoothness measures, two of first-order and two of second-order type, we reported on experiments showing that there is no support for the smoothness conjecture with respect to the first-order smoothness measures. However, our findings are quite well aligned with SGD enforcing second-order smoothness, thereby adding credibility to this instantiation of the conjecture.

For future work, it would be interesting to see if our results also transfer to deeper networks and larger datasets, as well as other network architectures, e.g. convolutional or recurrent networks. One could also now use theoretical tools to determine which second-order smoothness measure exactly is being minimized and by what mechanism.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P.A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X.: Tensorflow: A system for large-scale machine learning. In: USENIX Symposium on Operating Systems Design and Implementation, (OSDI) (2016)
2. Advani, M.S., Saxe, A.M.: High-dimensional dynamics of generalization error in neural networks. CoRR abs/1710.03667 (2017), http://arxiv.org/abs/1710.03667
3. Bartlett, P.L., Foster, D.J., Telgarsky, M.: Spectrally-normalized margin bounds for neural networks. In: Conference on Neural Information Processing Systems (NeurIPS) (2017)
4. Belkin, M., Hsu, D., Ma, S., Mandal, S.: Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proceedings of the National Academy of Sciences 116(32) (2019)
5. Caruana, R., Lawrence, S., Giles, C.L.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In: Conference on Neural Information Processing Systems (NeurIPS) (2000)
6. Drucker, H., Le Cun, Y.: Improving generalization performance using double backpropagation. IEEE Transactions on Neural Networks (T-NN) 3(6), 991–997 (1992)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Conference on Uncertainty in Artificial Intelligence (AISTATS) (2010)
8. Greff, K., Klein, A., Chovanec, M., Hutter, F., Schmidhuber, J.: The sacred infrastructure for computational research. In: Proceedings of the Python in Science Conferences-SciPy Conferences (2017)
9. Holopainen, R.: Smoothness under parameter changes: derivatives and total variation. In: Sound and Music Computing Conference (SMC) (2013)
10. Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., Bengio, S.: Fantastic generalization measures and where to find them. In: International Conference on Learning Representations (ICLR) (2020)
11. Kawaguchi, K., Kaelbling, L.P., Bengio, Y.: Generalization in deep learning. CoRR abs/1710.05468 (2017), http://arxiv.org/abs/1710.05468
12. Kendall, M.G.: A new measure of rank correlation. Biometrika 30(1/2), 81–93 (1938)
13. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: International Conference on Learning Representations (ICLR) (2017)
14. Kubo, M., Banno, R., Manabe, H., Minoji, M.: Implicit regularization in over-parameterized neural networks. CoRR abs/1903.01997 (2019), http://arxiv.org/abs/1903.01997
15. Lawrence, S., Gilesyz, C.L., Tsoi, A.C.: What size neural network gives optimal generalization? convergence properties of backpropagation. Tech. rep., Institute for Advanced Computer Studies, University of Maryland (1996)
16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
17. LeJeune, D., Balestriero, R., Javadi, H., Baraniuk, R.G.: Implicit rugosity regularization via data augmentation. CoRR abs/1905.11639 (2019), http://arxiv.org/abs/1905.11639

18. Liang, T., Poggio, T.A., Rakhlin, A., Stokes, J.: Fisher-rao metric, geometry, and complexity of neural networks. In: Conference on Uncertainty in Artificial Intelligence (AISTATS) (2019)
19. Maennel, H., Bousquet, O., Gelly, S.: Gradient descent quantizes ReLU network features. CoRR abs/1803.08367 (2018), http://arxiv.org/abs/1803.08367
20. Nagarajan, V., Kolter, J.Z.: Generalization in deep networks: The role of distance from initialization. In: Conference on Neural Information Processing Systems (NeurIPS) (2019)
21. Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., Sutskever, I.: Deep double descent: Where bigger models and more data hurt. In: International Conference on Learning Representations (ICLR) (2020)
22. Neyshabur, B., Tomioka, R., Srebro, N.: In search of the real inductive bias: On the role of implicit regularization in deep learning. In: International Conference on Learning Representations (ICLR) (2014)
23. Neyshabur, B., Tomioka, R., Srebro, N.: Norm-based capacity control in neural networks. In: Workshop on Computational Learning Theory (COLT) (2015)
24. Novak, R., Bahri, Y., Abolafia, D.A., Pennington, J., Sohl-Dickstein, J.: Sensitivity and generalization in neural networks: an empirical study. In: International Conference on Learning Representations (ICLR) (2018)
25. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive auto-encoders: Explicit invariance during feature extraction. In: International Conference on Machine Learing (ICML) (2011)
26. Sokolic, J., Giryes, R., Sapiro, G., Rodrigues, M.R.D.: Robust large margin deep neural networks. IEEE Transactions on Signal Processing (T-SP) 65(16), 4265–4280 (2017)
27. Spigler, S., Geiger, M., d'Ascoli, S., Sagun, L., Biroli, G., Wyart, M.: A jamming transition from under- to over-parametrization affects generalization in deep learning. Journal of Physics A: Mathematical and Theoretical 52(47) (2019)
28. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations (ICLR) (2017)