Generalization Comparison of Deep Neural Networks via Output Sensitivity

Mahsa Forouzesh
Ecole Polytechnique Federale de Lausanne
Lausanne, Switzerland
mahsa.forouzesh@epfl.ch

Farnood Salehi
DisneyResearch|Studios
Zurich, Switzerland
farnood.salehi@disneyresearch.com

Patrick Thiran
Ecole Polytechnique Federale de Lausanne
Lausanne, Switzerland
patrick.thiran@epfl.ch

Abstract—Although recent works have brought some insights into the performance improvement of techniques used in state-of-the-art deep-learning models, more work is needed to understand their generalization properties. We shed light on this matter by linking the loss function to the output's sensitivity to its input. We find a rather strong empirical relation between the output sensitivity and the variance in the bias-variance decomposition of the loss function, which hints on using sensitivity as a metric for comparing the generalization performance of networks, without requiring labeled data. We find that sensitivity is decreased by applying popular methods which improve the generalization performance of the model, such as (1) using a deep network rather than a wide one, (2) adding convolutional layers to baseline classifiers instead of adding fully-connected layers, (3) using batch normalization, dropout and max-pooling, and (4) applying parameter initialization techniques.

Index Terms—deep neural networks, generalization, sensitivity, bias-variance decomposition

I. INTRODUCTION

In machine-learning tasks, the main challenge a network designer faces is to find a model that learns from the training data and that is able to predict the output of unseen data with high accuracy. The first part is quite easily achievable by current over-parameterized deep neural networks, but the second part, referred to as generalization, demands careful expert hand-tuning [1], [2]. Modern convolutional neural network (CNN) architectures that achieve state-of-the-art results in computer-vision tasks, such as ResNet [3] and VGG [4], attain high-generalization performance. Part of their success is due to recent advances in hardware and the availability of large amounts of data, but their generalization performance remains unequal. Therefore, knowing when and why some models generalize, still remain open questions to a large extent [5].

In this paper, by investigating the link between sensitivity and generalization, we get one step closer to understanding the generalization properties of deep neural networks. Our findings suggest a relation between the sensitivity metric, a measure of uncertainty of the output with respect to input perturbations, and the variance term in the bias-variance decomposition of the validation loss. This relation gives insight in the link between sensitivity and loss when the bias is small, not only for classification tasks, but also for regression tasks.

Leveraging this relation, we can use the sensitivity metric to examine which network is more prone to overfitting. Our numerical results suggest sensitivity as an appealing metric that captures the generalization improvements brought by a large class of architectures and techniques used in state-of-the-art models. In summary, we make the following contributions:

- We provide an approximate relation between sensitivity and generalization loss, via the relation between sensitivity and variance in the bias-variance decomposition of the loss. Our empirical results on state-of-the-art convolutional neural networks suggest a surprisingly strong match between experimental results and this (rather crude) approximation.
- We propose sensitivity as a promising architecture-selection metric and show that sensitivity, similarly to the test loss, promotes certain architectures compared to others. We in particular study the addition of convolutional layers versus fully-connected ones, and depth versus width. Sensitivity can potentially be used as a neural architecture search (NAS) tool, a priori (before training), to automate the architecture-design process.
- We provide an alternative explanation for the success of batch normalization in terms of sensitivity. We further give a new viewpoint on the performance improvement of dropout and max-pooling, as networks with these methods have a lower sensitivity alongside a lower generalization loss. We show that sensitivity retrieves the effectiveness of He and Xavier parameter initialization techniques.

II. RELATED WORK

To the best of our knowledge, [6] was the first study to suggest a possible relation between sensitivity and generalization in multi-layer perceptrons, where the numerical results were limited to synthetic data. Recently, [7] suggested bounding the generalization error of deep neural networks with the spectral norm of the input-output Jacobian matrix, a measure of output sensitivity to its inputs. Reference [8] empirically compares sensitivity, measured by the norm of the Jacobian of the *output* of the softmax layer, and the generalization gap for fully-connected neural networks in image-classification tasks, leaving more complex architectures and other machine learning tasks as future work. Our empirical results presented in Section IV-C, together with the computations in Section IV-B, suggest that sensitivity *before* the softmax layer is related

to the generalization loss, and that computing the sensitivity before (as in our paper) or after (as in [8]) the softmax layer makes a strong difference (see e.g., Fig. 12 in the appendix). In our work, we elaborate on the relation between sensitivity and loss for a wide range of settings, beyond fully-connected networks and image-classification tasks. We also show a rather strong match between the expression computed in Section IV-B and the experimental results on state-of-the-art models.

To avoid overfitting in deep-learning architectures, regularization techniques are applied, such as weight decay, early stopping, dropout [9], and batch normalization (BN) [10]. A popular explanation for the improved generalization of dropout is that it combines exponentially many networks to prevent overfitting [9]. Reference [10] argues that the reason for the success of BN is that it addresses the internal-covariant-shift phenomenon. However, [11] argues against this belief and explains that the success of BN is due to its ability to make the optimization landscape smoother. In this paper, we look at the success of dropout and BN from another perspective: These methods decrease the output sensitivity to random input perturbations in a same manner as they decrease the test loss, resulting in better generalization performance.

Designing neural network architectures is one of the main challenges in machine-learning tasks. One major line of work in this regard compares deep and shallow networks [4], [12]–[16]. It is shown in [17] that to approximate a deep network, a shallow network requires an exponentially larger number of units per layer. After finding a satisfactory architecture, the trainability of the network needs to be carefully assessed. To avoid exploding or vanishing gradients, [18] and [19] introduce parameter initialization techniques that are widely used in current frameworks. By linking sensitivity and generalization, we present a new viewpoint on understanding the success of current state-of-the-art architectures and initialization techniques.

Previous theoretical studies attempting to explain the reasons for good generalization include generalization error (GE) bounds that use complexity measures such as VC-dimension and Rademacher complexities [20]. Encouraged by the ability of neural networks to fit an entire randomly labeled dataset [21], studies on data-dependent GE bounds have recently emerged [22]–[24]. Computing a practical non-vacuous GE bound that completely captures the generalization properties of deep neural networks is still an evolving area of research [5], [25], [26]. In this paper, we do not study GE bounds. We propose sensitivity as a practical proxy for generalization in a large number of settings.

There has been research on sensitivity analysis in neural networks with sigmoid and tanh activation functions [6], [27], [28]. Reference [29] introduces a sensitivity-based ensemble approach which selects individual networks with diverse sensitivity values from a pool of trained networks. Reference [30] performs a sensitivity analysis in neural networks to determine the required precision of the weight updates in each iteration. In this work, we extend these results to networks with

ReLU non-linearity with a different goal, which is to study the relation between sensitivity and generalization in state-of-the-art deep neural networks. Moreover, we provide a link between sensitivity and the variance in the bias-variance decomposition of the loss function.

There have been recent attempts to predict the test loss for supervised-learning tasks [8], [31], [32]. Reference [33] studies the module criticality, which is a weighted average over the distance of the network parameter vectors from their initial values. Although there seems to be a positive correlation between module criticality and generalization among different architectures, the correlation becomes negative when comparing the same architecture with different widths (as reported in Table 4 in [33]). Reference [34] introduces the so-called non-linearity coefficient (NLC) as a gradient-based complexity measure of the neural network, which is empirically shown to be a predictor of the test error for fully-connected neural networks. According to our results on both fully-connected and convolutional neural networks, sensitivity predicts the test loss, even before the networks are trained, which suggests sensitivity as a computationally inexpensive architecture-selection metric. Among the mentioned metrics, the Jacobian norm, studied in [8], does not require the computation of the parameter gradients nor the storage of large parameter vectors, as our metric, and therefore we compare it to sensitivity in Table I.

Paper Outline. We formally define loss and sensitivity metrics in Section III. In Section IV, we state the main findings of the paper and present the numerical and analytical results supporting them. Later in Section V, we propose a possible proxy for generalization properties of certain structures and certain methods and present the empirical results for a regression task with the Boston housing dataset. Finally in Section VI, we further discuss the observations followed up by a conclusion in Section VII. The empirical results for image-classification tasks presented in the main part of the paper are on the CIFAR-10 dataset and the empirical results for MNIST and CIFAR-100 datasets are deferred to Appendix F.

III. PRELIMINARIES

Consider a supervised-learning task, where the model predicts a ground-truth output $y \in \mathcal{Y} \subseteq \mathbb{R}^K$ for an input $x \in \mathcal{X} \subseteq \mathbb{R}^D$. The predictor $F_\theta : \mathcal{X} \to \mathcal{Y}$ is a deep neural network parameterized by the parameter vector θ that is learned on the training dataset $\mathcal{D}_t$ by using the stochastic learning algorithm $\mathcal{A}$. The training dataset $\mathcal{D}_t$ and the testing (validation) dataset $\mathcal{D}_v$ consist of i.i.d. samples drawn from the same data distribution p. With some abuse of notation, we use $\sim$ when the samples are uniformly drawn from a set of samples or from a probability distribution. We denote the $D \times D$ identity matrix by I_D.

A. Loss

Our main focus is a classification task where the loss function is the cross-entropy [35]. The average test loss can
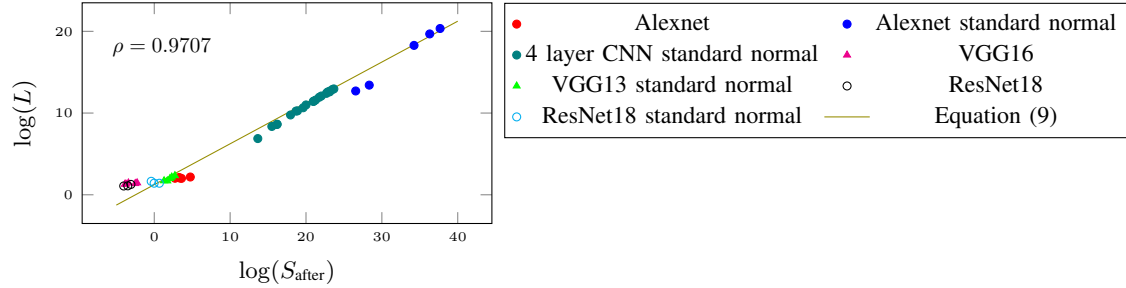
Fig. 1. Test loss L versus sensitivity S_{after} for popular CNN architectures. The parameter initialization is Xavier [18] with uniform distribution, unless stated as standard normal distribution. The networks are trained on a subset of the CIFAR10 training dataset and are evaluated on the entire CIFAR10 testing dataset. Each point of the plot indicates a network with a different number of channels and hidden units, and its coordinates $\log(L)$ and $\log(S_{\text{after}})$ are averaged over 10 runs. For more details on configurations refer to Appendix A. The Pearson correlation coefficient ρ between the data points is 0.9707.

be defined as

$$L = \mathbb{E}_{\theta^*}[L_{\theta^*}] = \mathbb{E}_{\theta^*}\left[\mathbb{E}_{(x,y)\sim\mathcal{D}_v}\left[-\sum_{k=1}^{K} y^k \log F_{\theta^*}^k(x)\right]\right], \tag{1}$$

where θ^* is the random[1] parameter vector found by $\mathcal{A}$, which minimizes the training loss defined on $\mathcal{D}_t$; K is the number of classes and $F_{\theta^*}^k$ is the k-th entry of the vector F_{θ^*}, which is the output of the softmax layer, i.e., $F_{\theta^*}(x) = \text{softmax}(f_{\theta^*}(x))$, where $f_{\theta^*}(x)$ is the output of the last layer of the network. In classification tasks, the neural network output is the probability assigned to each class and the output space is $\mathcal{Y} := [0,1]^K$.

B. Sensitivity

Let us inject an external noise to the input of the network and compute the resulting noise in the output before the softmax layer. The original input vector is $x \in \mathcal{X}$ to which we add an i.i.d. normal noise vector $\varepsilon_x \sim \mathcal{N}(0, \sigma_{\varepsilon_x}^2 I_D)$, then the output noise due to $\varepsilon_x \in \mathcal{X}$ is $\varepsilon_y = f_\theta(x + \varepsilon_x) - f_\theta(x)$.[2] We use the variance of the output noise, averaged over its K entries, as a measure of sensitivity: $S_\theta = \text{Var}(\overline{\varepsilon_y})$. The average sensitivity is therefore

$$S = \mathbb{E}_\theta[S_\theta] = \mathbb{E}_\theta\left[\text{Var}_{x,\varepsilon_x}\left[\frac{1}{K}\sum_{k=1}^{K}\varepsilon_y^k\right]\right], \tag{2}$$

where ε_y^k is the k-th entry of the vector ε_y. To distinguish the sensitivity S computed on untrained networks from trained ones, we denote $S_{\text{before}} = \mathbb{E}_\theta[S_\theta]$ and $S_{\text{after}} = \mathbb{E}_{\theta^*}[S_{\theta^*}]$ when the expectation is over the network parameters before and after training, respectively. We consider an "unspecific" sensitivity (meaning that the average is taken over all the entries of the output noise), which requires unlabeled data samples, as opposed to the "specific" sensitivity (which is limited to the output of the desired class) defined in [36]. In this work, the input vectors x used for computing S are drawn from $\mathcal{D}_v$, so that given a new test data point, the sensitivity S predicts which trained network performs better for this particular

point, and therefore gives a real-time uncertainty metric for predicting unseen data. For a few network architectures, we computed S on the training set $\mathcal{D}_t$ and observed that its value is practically the same as S computed on the testing set $\mathcal{D}_v$ (see Fig. 13 in the appendix), which suggests that S as a generalization metric does not require sacrificing a set of training points for validation.

IV. SENSITIVITY VERSUS LOSS

A. Numerical Experiments

A good predictor should give similar outputs to similar inputs in most cases. Assuming that an unseen data point is drawn from the same distribution as an training data point, the two concepts of robustness (insensitivity) and generalization should therefore be related. Robustness here is the average-case robustness, not the worst-case robustness (adversarial robustness). We measure it by computing S (Equation (2)), and considering near-zero training loss, we refer to the test loss L (Equation (1)) as the generalization error. According to our observations on a wide set of experiments, including ResNets [3] and VGGs [4], we find a rather strong relation between S and L. State-of-the-art networks decrease the generalization error alongside with the sensitivity of the output of the network with respect to the input (Fig. 1).

Many factors influence the generalization performance of deep-learning models, among which network topology, initialization technique, and regularization method. In Section V, we study the influence of each of these three factors on S and keep all the other factors, including the learning algorithm, the same throughout the experiments. These experiments suggest the use of S_{after} as a proxy to the test loss, which is particularly advantageous for settings where labeled training data is limited; assessing generalization performance can then be done without having to sacrifice training data for the validation set. Furthermore, S_{before} can potentially be used as an architecture-selection metric before training the models. We refer to fully-connected neural networks as FC, and to convolutional neural networks as CNN.

B. Bias-Variance Decomposition

In this section, a crude approximate relation between sensitivity S and generalization error L is established through

---

[1]The randomness is introduced by the stochastic optimization algorithm $\mathcal{A}$ and the randomized parameter initialization technique.

[2]To reconstruct the vector $x + \varepsilon_x$ into an image a normalization might be needed, but this is unnecessary when $x + \varepsilon_x$ is the neural network input.

the link between sensitivity and the variance term in the bias-variance decomposition of the mean square error (MSE). First, we find the link between the cross-entropy loss and MSE. Next, we develop the relation between sensitivity and the variance term, and finally, the link between S and L.

When the predictor $F_{\theta*}(x)$ assigns the probability $F_{\theta*}^c(x)$ to the correct class c and $1 - F_{\theta*}^c(x)$ to another class (see Appendix C for details), the cross-entropy loss L can be approximated as[3]

$$L \approx \mathbb{E}_{(x,y,\theta*)}\left[\frac{1}{\sqrt{2}}\sqrt{\sum_{k=1}^{K}\left(F_{\theta*}^k(x) - y^k\right)^2}\right]. \quad (3)$$

We roughly approximate the right-hand side in (3) with $\sqrt{L_{\text{MSE}}/2}$, where L_{MSE} is the mean square error criterion defined as

$$L_{\text{MSE}} = \mathbb{E}_{\theta*}[L_{\theta*\text{MSE}}] = \mathbb{E}_{\theta*}\left[\mathbb{E}_{(x,y)\sim\mathcal{D}_v}\left[\|F_{\theta*}(x) - y\|_2^2\right]\right]. \quad (4)$$

Consider the classic notion of bias-variance decomposition for the MSE loss [37]–[40], where the generalization error is the sum of three terms: bias, variance and noise, i.e., $L_{\text{MSE}} = \varepsilon_{\text{bias}} + \varepsilon_{\text{variance}} + \varepsilon_{\text{noise}}$. In this work, we consider the label-collection process to be noiseless and neglect the third term $\varepsilon_{\text{noise}}$. The bias term is formally defined as

$$\varepsilon_{\text{bias}} = \mathbb{E}_{x,y}\left[\|\mathbb{E}_{\theta*}[F_{\theta*}(x)] - y\|_2^2\right], \quad (5)$$

and the variance term is

$$\varepsilon_{\text{variance}} = \sum_{k=1}^{K}\mathbb{E}_x\left[\text{Var}_{\theta*}(F_{\theta*}^k(x))\right]. \quad (6)$$

Let us now draw an again crude approximate relation between $\varepsilon_{\text{variance}}$ and S under strong assumptions on the probability distributions of θ, x, and ε_x (refer to Appendix B for more details). Given a feed-forward neural network with M hidden layers and H_l units per layer, $1 \leq l \leq M$, where the non-linear activation function is positive homogeneous[4] with parameters α and β (Equation (11) in Appendix B), we have

$$\varepsilon_{\text{variance}} \approx \left(\frac{K-1}{K}\right)\left(S \cdot \frac{\sigma_x^2}{\sigma_{\varepsilon_x}^2} + \Sigma\right), \quad (7)$$

where

$$\Sigma = \frac{1}{K}\sum_{l=1}^{M}\sigma_{b_l}^2\prod_{i=l+1}^{M}\left(\frac{\alpha^2+\beta^2}{2}\right)\sigma_{w_i}^2 H_i, \quad (8)$$

where K is the number of units in the output of the softmax layer and $\sigma_{w_l}^2$, $\sigma_{b_l}^2$, σ_x^2, and $\sigma_{\varepsilon_x}^2$ are the second moment of weights and biases of layer l, input x and input noise ε_x, respectively. Equation (8) can be extended to convolutional neural networks[5] by replacing H_i with fan_{in} of layer i.

Given an infinite amount of training data, the bias represents the best performance of the model, which can be approximated

[3]This is more accurate for over-confident predictors (see Appendix C).
[4]ReLU is a positive homogeneous function with $\alpha = 1$ and $\beta = 0$.
[5]fan_{in} = the number of input channels $*$ the kernel size

TABLE I
PEARSON'S CORRELATION COEFFICIENT ρ BETWEEN EACH METRIC (S, J) AND THE TEST LOSS (L), AND AVERAGE COMPUTATION TIME (IN SECONDS) OF EACH METRIC FOR VGG13, VGG16, RESNET18 AND RESNET34 NETWORKS TRAINED ON THE CIFAR-10 DATASET. THE TEST ACCURACY OF THE NETWORKS ARE UP TO 87%.

Metric	ρ	Computation time
J after softmax	0.116	1.166 ± 0.111
J before softmax	0.414	1.165 ± 0.111
S after softmax	0.381	0.086 ± 0.006
S before softmax	**0.648**	**0.085** ± 0.006

by the training loss [40]–[42]. In deep learning settings (and thus in our experiments), the training loss is close to zero, hence if we neglect the bias term $\varepsilon_{\text{bias}}$ in the decomposition of L_{MSE} we have

$$L \approx \sqrt{\frac{1}{2}\left(\frac{K-1}{K}\right)\left(S \cdot \frac{\sigma_x^2}{\sigma_{\varepsilon_x}^2} + \Sigma\right)}, \quad (9)$$

where Σ is given by (8). In the experiments, we observe that $\sigma_{b_l}^2$ is usually very small or zero (for instance in ResNets because $b_l = 0$), making $\Sigma \approx 0$.

According to (7) and the relation between L_{MSE} and L, to compare networks with a small value of $\varepsilon_{\text{bias}}$ (which is usually the case in deep neural networks where the bias is approximated by the near-zero training loss), the test loss can be approximated using the sensitivity by (9). Despite the strong assumptions and crude approximations to get (9), the numerical experiments show a rather surprisingly good match with (9) (Figs. 1, 2 and 3), even if Σ is neglected in (9). It is interesting to note that the right-hand side of (9) is computed without requiring labeled data points, whereas the left-hand side requires the ground-truth output vector y.

If $\varepsilon_{\text{bias}}$ can no longer be approximated by the training loss, which may in part explain the poorer match in lower values of S_{after} in Fig. 1, we need more training data to make this approximation valid. In Section VI-A we train the networks with more data samples and observe that numerical results become closer to (9).

C. Sensitivity Before Versus After The Softmax Layer

It is interesting to compare the sensitivity S given by (2) with the Frobenius norm of the Jacobian matrix J of the output of the softmax layer [8], in terms of their ability to gauge the generalization error L. A practical motivation for using S instead of J in real-world applications is computational tractability: to find the network architecture(s) with the best generalization ability among a collection of trained networks, the computation of S does not require to make a backward pass through each network architecture, contrary to J. In Table I we observe that computing Jacobian is more than 10 times slower than computing sensitivity. But the main motivation for using S is that it is computed *before* and not *after* the softmax layer, contrary to J in [8]. Because of the chain rule, J depends on the derivative of the softmax function
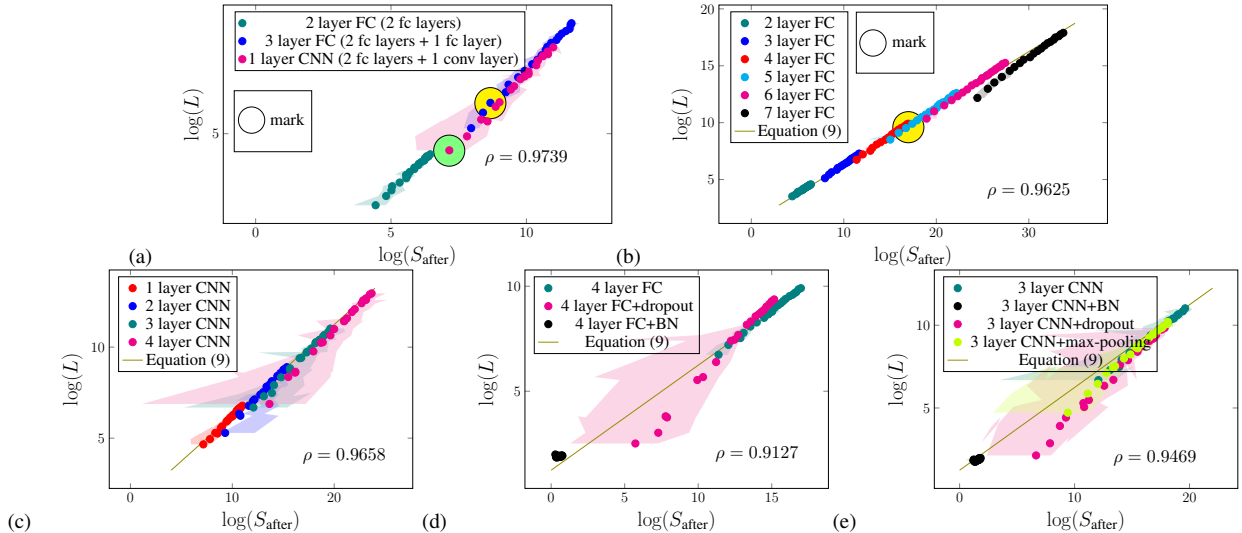
Fig. 2. Test loss L versus sensitivity S_{after} for networks trained on a subset of the CIFAR-10 training dataset where the network parameters are initially drawn from a standard normal distribution. Each point of the plot indicates a network with a different number of channels and hidden units, and its coordinates $\log(L)$ and $\log(S_{\text{after}})$ are averaged over 10 runs. The shaded areas are contained by the minimum and maximum values of $\log(S_{\text{after}})$ over multiple runs of each experiment (point). **(a)** Comparison between adding a convolutional layer and adding a fully-connected layer to a baseline classifier that is a fully-connected neural network with one hidden layer. **(b)** Fully-connected neural networks. **(c)** Convolutional neural networks. **(d)** 4-layer FC trained with or without regularization. **(e)** 3-layer CNN trained with or without regularization. The relation between $\log(L)$ and $\log(S)$ is linear according to (9), unlike the relation between S and L, hence a log-log scale has been chosen for all the figures.

with respect to the logits, which has very low values for highly confident predictors (the ones which assign a very high probability to one class and almost zero probability to the other classes). For instance, if the predictor erroneously assigns a high probability to a wrong class, the derivative of the softmax function is very low, resulting in a very low J. In this case, J would be misleading as it would mistakenly indicate good generalization. In contrast, S does not depend on the confidence level of the predictor. The difference is illustrated in Table I (see also Fig. 12 in the appendix), where the strong correlation between S and L (and the good match with (9)) is not found when S is replaced by the sensitivity *after* the softmax layer. Therefore, we observe from Table I that S computed before the softmax layer (given by (2)), is preferred to J (defined in [8]), both in terms of correlation to the test loss and of computation time.

V. Sensitivity as a Proxy for Generalization

In this section, we argue that methods improving the generalization performance of neural networks remarkably reduce the sensitivity S. We also present the experimental results for a regression task.

A. Comparing Different Architectures

Convolutional vs Fully-Connected Layers. The relation between the sensitivity S and the generalization error L supports the common view that CNNs outperform FCs in image-classification tasks. In Fig. 2 (a) we empirically observe that, given a CNN and an FC with the same number of parameters, the CNN has lower sensitivity and test loss than the FC. Moreover, some CNNs with more parameters than FCs have both lower sensitivity and lower test loss, even though

they are more over-parameterized.

Let us start from a baseline classifier with one hidden layer (2 layers in total displayed in teal blue points in Fig. 2 (a), where each point represents a network with a different number of hidden units). We compare the effect of adding another fully-connected layer with adding a convolutional layer in Fig. 2 (a). We vary the number of parameters of 1-layer CNNs (which consist of 2 fully-connected (fc) layers and 1 conv layer, displayed by pink points) from 450k to 10M by increasing the number of channels and hidden units, whereas the number of parameters for 3-layer FCs varies from 320k to 1.7M (displayed by dark blue points). Despite the large number of parameters of CNNs, they suffer from less overfitting and have a lower sensitivity S than FCs. Next, let us compare a FC to a CNN with the same number of parameters in Fig. 2 (a): A 3-layer FC with 140 units in each layer (yellow mark) and a 1-layer CNN with 5 channels and 100 units (green mark), both have 450k parameters. The CNN has remarkably lower sensitivity and test loss than the FC, which indicates better performance compared to the FC with the same number of parameters.

Depth vs Width. Consider a feedforward FC with ReLU activation function where all the network parameters follow the standard normal distribution and are independent from each other and from the input. If we have M layers with H units in each hidden layer, K units in the output layer and D units in the input layer, then (see Appendix D for details)

$$S = \frac{D}{K} \left(\frac{H}{2} \right)^M \sigma_{\varepsilon_x}^2. \tag{10}$$

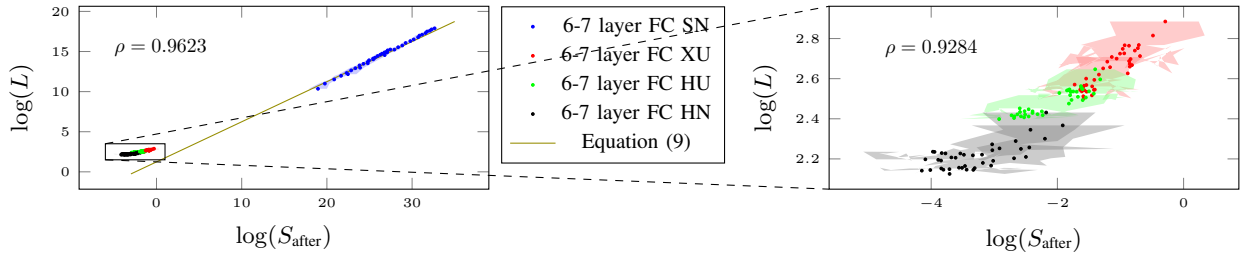According to (10), considering two neural networks with the same value for H^M, one deep and narrow (higher M and

Fig. 3. Test loss L versus sensitivity S_{after} for networks trained on a subset of the CIFAR-10 training dataset where networks are initialized with different methods. On the right, we have a zoom in plot of the bottom left frame of the left figure.

lower H), and the other one shallow and wide (lower M and higher H), the deeper network has lower sensitivity S. Assuming that both networks have near-zero training losses, depth is better than width regarding generalization in fully-connected neural networks. The empirical results in Fig. 2 (b) support (10). For instance, a 4-layer FC with 500 units per layer (the top right most point among all 4-layer FCs, indicated by a yellow mark), has the same value for $(H/2)^M$ as a 5-layer FC with 165 units per layer (the 4th point among 5-layer FCs, which exactly matches the yellow mark). In Fig. 2 (b), these two networks have the same values of both S_{after} and L, and all narrower 5-layer networks (with 100, 120, and 140 units) have better performance than the wide 4-layer network (with 500 units). A similar trend is observed for CNNs in Fig. 2 (c): having a narrower and deeper CNN is preferable to having a wider and shallower CNN.

B. Regularization Techniques

Figs. 2 (d) and (e) show the sensitivity S_{after} versus the test loss L, for different regularization methods. In particular, we study the effect of dropout [9] and batch normalization (BN) [10] on the sensitivity in the FCs; and we apply dropout, BN and max-pooling for the CNNs. The results are consistent with the relation between sensitivity S_{after} and loss L. For all these regularization techniques, we observe a shift of the points towards the bottom left. This shift shows that these techniques known to improve generalization simultaneously decrease the network sensitivity to input perturbations. This is particularly noticeable in the BN case, where both the sensitivity and test loss decrease dramatically. This suggests that batch normalization improves performance by making the network less sensitive to input perturbations.

C. Initialization Methods

Another interesting observation is the effect of various parameter initialization techniques on the sensitivity and loss values, after the networks are trained (Fig. 3). We consider four initialization techniques for network parameters in our experiments: (i) Standard Normal distribution (SN), (ii) Xavier [18] initialization method with uniform distribution (XU), (iii) He [19] initialization method with uniform distribution (HU), and (iv) He initialization method with normal distribution (HN). As shown in Fig. 3, the relation between sensitivity S_{after} and test loss L provides us with a new viewpoint on
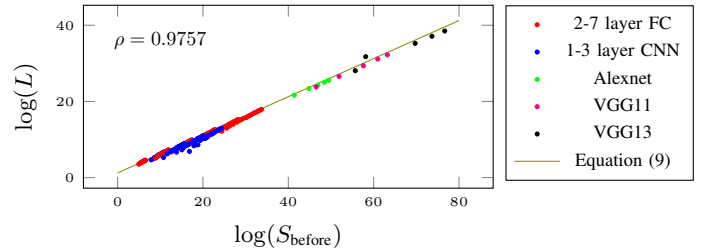

Fig. 4. Test loss of trained models, L, versus sensitivity of untrained models, S_{before}, for networks whose parameters are initially drawn from the standard normal distribution. Note that the regularization techniques BN, dropout and max-pooling are removed from Alexnet, VGG11, and VGG13 configurations.

the success of the state-of-the-art initialization techniques; HN has the best generalization performance, alongside the lowest sensitivity value (the black points in Fig. 3).

D. Sensitivity of Untrained Networks as a Proxy for Generalization Loss

A similar trend is observed for neural networks that are not yet trained. In Fig. 4, the sensitivity S_{before} is measured before the networks are trained, and the test loss L is measured after the networks are trained. The parameters in the fully-connected and convolutional networks are initialized by sampling from the standard normal distribution, and no explicit regularization (dropout, BN, max-pooling) is used in the training process. These two conditions are necessary, because regularization techniques only affect the training process, hence S_{before} is the same for networks with or without regularization layers, and the He and Xavier initialization techniques force the sensitivity to be the same regardless of the number of units in hidden layers. Therefore, under these two conditions, the generalization performance of untrained networks with different architectures can be compared. The strong link between the sensitivity of untrained networks S_{before} and the test loss L observed in Fig. 4 suggests that the generalization of neural networks can be compared before the networks are even trained, making sensitivity a computationally inexpensive architecture-selection method.

E. Regression Task and MSE Loss

In this section, we investigate the relation between sensitivity and generalization error for regression tasks with the mean square error criterion (MSE). The loss function in this setting
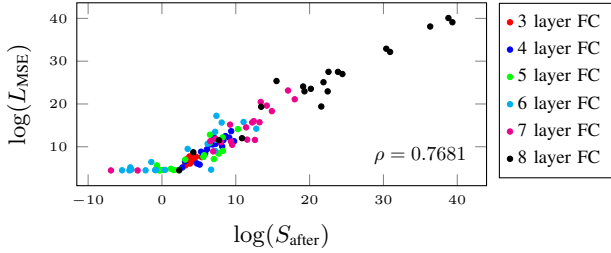
Fig. 5. Test loss L_{MSE} versus sensitivity S_{after} for a regression task with the MSE loss criterion. The networks are trained and evaluated on the Boston house price dataset. Each point of the plot indicates a network with a different width and its coordinates are averaged over 10 runs.

is defined in (4) where θ^* is found by minimizing MSE on training dataset $\mathcal{D}_t$ using the stochastic learning algorithm $\mathcal{A}$. Note that the output is the last layer of the neural network (the softmax layer is not applied), and that the output layer has 1 unit, i.e., $K = 1$, and that y is a scalar. The sensitivity is defined as $S = \mathbb{E}_{\theta^*}\left[\text{Var}_{x,\varepsilon_x}\left[F_{\theta^*}(x + \varepsilon_x) - F_{\theta^*}(x)\right]\right]$ and the bias and variance terms are defined by (5) and (6), respectively. We consider the Boston housing dataset where the objective is to predict the price of a house given 14 features (including crime rate, distance to employment centers, etc.). Fig. 5 shows sensitivity versus test loss among fully-connected neural networks with 3-8 layers and 100-500 hidden units per layer; the networks are trained on 70% of the dataset and then evaluated on the remaining 30%. The results are consistent with the relation between sensitivity S and generalization error, which for the regression task is L_{MSE}. For a more detailed view, we observe that sensitivity is related to the variance in the bias-variance decomposition of the MSE loss (Fig. 10 (d) in the appendix), and the MSE loss is the sum of bias and variance terms (Fig. 10 (c) in the appendix).

VI. Discussion

A. Discussion Regarding Bias

In this section, we discuss the role of the number of training samples and of the stage of the training on the validity of the approximation made in (9) where we neglect $\varepsilon_{\text{bias}}$. In our experiments, we observe that when the number of training samples is low (see for instance Fig. 11 (a) for ResNet18 and ResNet34 networks), the match between experiments and (9) is rather poor. We show (in Fig. 11 (b) in the appendix) that this problem can be solved (at least in part) by training the networks with more samples: for instance, in Fig. 11 (b) the yellow marks are ResNet18, the green marks are ResNet34, and the results show a relation between $\log S_{\text{after}}$ and $\log L$ that becomes linear as we add more training data samples in the training process. Therefore, the larger the number of training samples is, the better the approximation $\varepsilon_{\text{bias}} \approx trainloss$ becomes, and $\varepsilon_{\text{variance}}$ becomes the more dominant term in the test loss. We also observe that, when computing sensitivity and loss at different stages of training, the bias term $\varepsilon_{\text{bias}}$ in the test loss cannot be neglected at initial stages of the training. As the training progresses, the experimental results get closer to (9) (see Fig. 11 (d) in the appendix).

B. Final Remarks

As discussed in Section IV-B, the loss can be decomposed in three terms: $\varepsilon_{\text{variance}}$, $\varepsilon_{\text{bias}}$, and $\varepsilon_{\text{noise}}$. The proposed relation between sensitivity S and variance $\varepsilon_{\text{variance}}$ is extended to a relation between S and generalization loss L when $\varepsilon_{\text{variance}}$ is the dominant term in the decomposition of the loss, which is often the case in deep learning settings. In the previous section, we discussed the possibility that the bias term $\varepsilon_{\text{bias}}$ might not be negligible compared to $\varepsilon_{\text{variance}}$, when the number of training samples is low and when the training loss is large. When the available data contains randomly labeled samples, then $\varepsilon_{\text{noise}}$ can no longer be neglected. As S does not depend on the labels, the randomness in the labels, and therefore the generalization performance of the model, can no longer be entirely captured by sensitivity in this setting. Furthermore, the pixel-wise linear input perturbations considered in our experiments might not be realistic; ideally, we would like to perturb the input in the latent space of the generative model of the input image. Also, the relation between S and L requires the non-linearity to be positive homogeneous. The generalization properties of networks with sigmoid and tanh activation functions are left for future work.

The sensitivity S changes with input-output re-scaling: For a homogeneous predictor, if the input data scale is multiplied by a factor α, and the output is divided by the factor α, then L remains unchanged, whereas S gets divided by α^4. However, as long as we compare networks subject to the same input data distribution, this re-scaling obviously does not happen. Moreover, S can be affected by output re-scaling: If the output of a classifier is divided by a factor α, then the classification accuracy remains the same, whereas L and S get divided by (approximately) α and α^2, respectively. While the relation between L and S remains valid, there is a mismatch between accuracy and loss, which suggests that the networks are miscalibrated. Applying network calibration methods such as *temperature scaling* [43] can potentially increase the correlation between the cross-entropy loss L and the classification error (i.e., 1 - accuracy), as well as the correlation between the sensitivity S and the classification error (see e.g., Table II in the appendix).

The relation between sensitivity and variance can be extended to any loss that admits a bias-variance decomposition. Therefore, if such a decomposition is found for the classification error (which might not be purely additive [44]), which is still an active research topic, then the link between sensitivity and error would follow. We note that there is a difference between causality and correlation between a complexity measure and generalization, as discussed in [45]. We study the correlation between S and L, however, this does not imply that there is a causal relation between the two.

VII. Conclusion

We find that the sensitivity metric is a strong indicator of overfitting. Given multiple networks having near-zero training losses to choose from with different hyper-parameters, the best architecture appears to be the one with the lowest

sensitivity value. Sensitivity can also potentially be used as an architecture-selection method. One of the advantages of the sensitivity metric is that it can provide a loose prediction of the test loss without the use of any labeled data. This is especially important in applications where labeling data is expensive.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," in *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

[6] Y. Dimopoulos, P. Bourret, and S. Lek, "Use of some sensitivity criteria for choosing networks with good generalization ability," *Neural Processing Letters*, vol. 2, no. 6, pp. 1–4, 1995.

[7] J. Sokolić, R. Giryes, G. Sapiro, and M. R. Rodrigues, "Robust large margin deep neural networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4265–4280, 2017.

[8] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: an empirical study," in *International Conference on Learning Representations*, 2018.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[11] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," in *Advances in Neural Information Processing Systems*, pp. 2483–2493, 2018.

[12] Y. Bengio and O. Delalleau, "On the expressive power of deep architectures," in *International Conference on Algorithmic Learning Theory*, pp. 18–36, Springer, 2011.

[13] H. Mhaskar, Q. Liao, and T. Poggio, "When and why are deep networks better than shallow ones?," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[14] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[15] J. Ba and R. Caruana, "Do deep nets really need to be deep?," in *Advances in neural information processing systems*, pp. 2654–2662, 2014.

[16] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in neural information processing systems*, pp. 2924–2932, 2014.

[17] M. Telgarsky, "Benefits of depth in neural networks," *arXiv preprint arXiv:1602.04485*, 2016.

[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[20] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[21] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[22] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, "Generalization in deep learning," *arXiv preprint arXiv:1710.05468*, 2017.

[23] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

[24] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang, "Stronger generalization bounds for deep nets via a compression approach," *arXiv preprint arXiv:1802.05296*, 2018.

[25] G. K. Dziugaite and D. M. Roy, "Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data," *arXiv preprint arXiv:1703.11008*, 2017.

[26] V. Nagarajan and J. Z. Kolter, "Uniform convergence may be unable to explain generalization in deep learning," in *Advances in Neural Information Processing Systems*, pp. 11615–11626, 2019.

[27] L. Fu and T. Chen, "Sensitivity analysis for input vector in multilayer feedforward neural networks," in *Neural Networks, 1993., IEEE International Conference on*, pp. 215–218, IEEE, 1993.

[28] X. Zeng and D. S. Yeung, "Sensitivity analysis of multilayer perceptron to input and weight perturbations," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1358–1366, 2001.

[29] J. Yang, X. Zeng, S. Zhong, and S. Wu, "Effective neural network ensemble approach for improving generalization performance," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 6, pp. 878–887, 2013.

[30] S. W. Piche, "The selection of weight accuracies for madalines," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 432–445, 1995.

[31] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio, "Predicting the generalization gap in deep networks with margin distributions," *arXiv preprint arXiv:1810.00113*, 2018.

[32] T. E. Wang, J. Gu, D. Mehta, X. Zhao, and E. A. Bernal, "Towards robust deep neural networks," *arXiv preprint arXiv:1810.11726*, 2018.

[33] N. S. Chatterji, B. Neyshabur, and H. Sedghi, "The intriguing role of module criticality in the generalization of deep networks," *arXiv preprint arXiv:1912.00528*, 2019.

[34] G. Philipp and J. G. Carbonell, "The nonlinearity coefficient-predicting overfitting in deep neural networks," *arXiv preprint arXiv:1806.00179*, 2018.

[35] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*, pp. 499–515, Springer, 2016.

[36] E. Tartaglione, S. Lepsøy, A. Fiandrotti, and G. Francini, "Learning sparse neural networks via sensitivity-driven regularization," in *Advances in Neural Information Processing Systems*, pp. 3878–3888, 2018.

[37] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[38] R. Tibshirani, *Bias, variance and prediction error for classification rules*. Citeseer, 1996.

[39] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas, "A modern take on the bias-variance tradeoff in neural networks," *arXiv preprint arXiv:1810.08591*, 2018.

[40] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists," *Physics reports*, 2019.

[41] A. Ng, "Learning Thoery Lecture Notes." http://cs229.stanford.edu/notes/cs229-notes4.pdf.

[42] S. Fortmann-Roe, "Understanding the bias-variance tradeoff." http://scott.fortmann-roe.com/docs/BiasVariance.html, 2012.

[43] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *arXiv preprint arXiv:1706.04599*, 2017.

[44] P. Domingos, "A unified bias-variance decomposition," in *Proceedings of 17th International Conference on Machine Learning*, pp. 231–238, 2000.

[45] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio, "Fantastic generalization measures and where to find them," *arXiv preprint arXiv:1912.02178*, 2019.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[47] I. Bellido and E. Fiesler, "Do backpropagation trained neural networks have normal weight distributions?," in *International Conference on Artificial Neural Networks*, pp. 772–775, Springer, 1993.