

Visualizing high-dimensional loss landscapes with Hessian directions

Lucas Böttcher^{1,2,*} and Gregory Wheeler^{1,†}

¹*Department of Computational Science and Philosophy,
Frankfurt School of Finance and Management, Frankfurt am Main, 60322, Germany*

²*Department of Medicine, University of Florida, Gainesville 32610, FL, USA*

(Dated: December 5, 2023)

Analyzing geometric properties of high-dimensional loss functions, such as local curvature and the existence of other optima around a certain point in loss space, can help provide a better understanding of the interplay between neural network structure, implementation attributes, and learning performance. In this work, we combine concepts from high-dimensional probability and differential geometry to study how curvature properties in lower-dimensional loss representations depend on those in the original loss space. We show that saddle points in the original space are rarely correctly identified as such in expected lower-dimensional representations if random projections are used. The principal curvature in the expected lower-dimensional representation is proportional to the mean curvature in the original loss space. Hence, the mean curvature in the original loss space determines if saddle points appear, on average, as either minima, maxima, or almost flat regions. We use the connection between expected curvature in random projections and mean curvature in the original space (*i.e.*, the normalized Hessian trace) to compute Hutchinson-type trace estimates without calculating Hessian-vector products as in the original Hutchinson method. Because random projections are not suitable to correctly identify saddle information, we propose to study projections along dominant Hessian directions that are associated with the largest and smallest principal curvatures. We connect our findings to the ongoing debate on loss landscape flatness and generalizability. Finally, for different common image classifiers and a function approximator, we show and compare random and Hessian projections of loss landscapes with up to about 7×10^6 parameters.

I. INTRODUCTION

Every deep neural network loss function depends on parameters, $\theta \in \mathbb{R}^N$, which induce a loss landscape that is typically in high dimensions [1]. The loss landscape associated with a neural network and the performance of optimization procedures that operate on it are each influenced by several factors, including the structural properties of the neural network [2–4] and a range of implementation attributes one may choose [5–9]. The exact nature of these factors and how their combinations influence learning performance, however, remain largely unknown.

One way to better understand the interplay between neural network structure, implementation attributes, and learning performance is through a better understanding of the geometric properties of loss landscapes. For example, Keskar *et al.* [10] analyze the local curvature around candidate minimizers via the spectrum of the underlying Hessian to characterize the flatness and sharpness of loss minima, and Dinh *et al.* [11] demonstrate that reparameterizations can render flat minima sharp without affecting generalization properties. Another approach, which holds out the promise of visualizing the curvature around a point and the existence, if any, of nearby optima, aims to visualize high-dimensional loss functions by a lower-dimensional (and often random) projection of two or three dimensions [12–16]. Building on this approach, Horoi *et al.* [16] pursue improvements in learning by dynamically sampling points in projected low-loss regions surrounding local minima during training.

Given the advantages afforded by the promise of visualizing loss landscape optima, one may ask: how do the curvature properties in low-dimensional visualizations of high-dimensional functions depend on the curvature properties of the original, high-dimensional loss space? Specifically, do random two- and three-dimensional loss projections meaningfully represent convexity and concavity properties of high-dimensional loss functions? In short, they do not. Saddle points are generally misrepresented in the expected lower-dimensional representation of an original, high dimensional loss space. Yet, due to the exponential proliferation of saddle points in high dimensions [17, 18], a critical point in high-dimensional loss space is almost certain to be a saddle rather than a minimum. Nevertheless, random projections *can* be useful to obtain curvature estimates, as we shall demonstrate by establishing a connection between expected curvature in random loss projections and Hessian trace estimates based on Hutchinson’s method [19].

Our contributions in this paper are threefold. First, we explain why random projections from a high-dimensional space do not preserve curvature information. This argument is developed in Section II. One consequence of this

* lbottcher@ufl.edu

† g.wheeler.de

result is that saddle points of an original high-dimensional loss landscape are not identified correctly in the associated expected low-dimensional random projection, a point we illustrate with simulation examples in Section III A. Second, while principal curvatures associated with a random projection are not weighted ensemble means of the principal curvatures in the original high-dimensional space, the principal curvatures in a low-dimensional projection *are* given by functions of weighted ensemble means of the Hessian elements in the original, high-dimensional space. This result is developed in Section II as well, and it is also illustrated by a simulation example in Section III A. A complementary simulation example in Section III B shows how one can estimate mean curvature from the slope of random loss projections. Together, Section II is intended to be a self-contained description of the relationship between principal curvatures of a loss function and the principal curvatures associated with a lower-dimensional, random projection.

Our third contribution builds on the results reported in Sections II and III A to propose an efficient method for estimating and visualizing Hessian information in high-dimensional loss landscapes. Instead of using random projections to visualize loss functions, we propose in Sections III C and IV to analyze projections along *dominant Hessian directions* that are associated with the largest-magnitude positive and negative principal curvatures, *i.e.*, those directions along which the magnitude of positive and negative changes of the loss function is largest. For different common image classifiers and a function approximator as studied in [20], we show and compare random and Hessian projections of loss landscapes with up to about 7×10^6 parameters. In accordance with related works unlocking Hessian information in non-linear and high-dimensional settings [21, 22], our proposal uses Hessian-vector products (HVPs), thereby bypassing the computational expense of computing a Hessian matrix. In short, the role of random projections in our approach is to estimate mean curvature and weighted ensemble means of elements of the original high-dimensional Hessian.

Finally, we conclude our paper in Section V with a discussion of our results. Our source code is publicly available at [23].

II. PRINCIPAL CURVATURE IN RANDOM PROJECTIONS

To describe the connection between the principal curvature of a loss function $L(\theta)$ and that associated with a lower-dimensional, random projection, we provide in Section II A an overview of concepts from differential geometry [24–26] that are useful to mathematically describe curvature in high-dimensional spaces. In Sections II B and II C, we show the relationship between principal curvature and random projections. Finally, in Section II D, we highlight a relationship between measures of curvature presented in Section II C and Hutchinson-type Hessian trace estimates.

A. Differential and information geometry concepts

In differential geometry, the principal curvatures are the eigenvalues of the shape operator (or Weingarten map¹), a linear endomorphism defined on the tangent space T_p of L at a point p . For the original high-dimensional space, we have $(\theta, L(\theta)) \subseteq \mathbb{R}^{N+1}$ and there are N principal curvatures $\kappa_1^\theta \geq \kappa_2^\theta \geq \dots \geq \kappa_N^\theta$. At a non-degenerate critical point θ^* where the gradient $\nabla_\theta L$ vanishes, the matrix of the shape parameter is given by the Hessian H_θ with elements $(H_\theta)_{ij} = \partial^2 L / (\partial \theta_i \partial \theta_j)$ ($i, j \in \{1, \dots, N\}$).² Some works refer to the Hessian as the “curvature matrix” [28] or use it to characterize the curvature properties of $L(\theta)$ [13]. In the vicinity of a non-degenerate critical point θ^* , the eigenvalues of the Hessian H_θ are the principal curvatures and describe the loss function in the eigenbasis of H_θ according to

$$L(\theta^* + \Delta\theta) = L(\theta^*) + \frac{1}{2} \sum_{i=1}^N \kappa_i^\theta \Delta\theta_i^2. \quad (1)$$

The Morse lemma states that, if a critical point θ^* of $L(\theta)$ is non-degenerate, then there exists a chart $(\tilde{\theta}_1, \dots, \tilde{\theta}_N)$ in a neighborhood of θ^* such that

$$L(\tilde{\theta}) = -\tilde{\theta}_1^2 - \dots - \tilde{\theta}_i^2 + \tilde{\theta}_{i+1}^2 + \dots + \tilde{\theta}_N^2 + L(\theta^*), \quad (2)$$

where $\tilde{\theta}_i(\theta^*) = 0$ for $i \in \{1, \dots, N\}$. The loss function $L(\tilde{\theta})$ in Eq. (2) is decreasing along i directions and increasing along the remaining $i + 1$ to N directions. Further, the index i of a critical point θ^* is the number of linearly

¹ Some authors distinguish between the shape operator and the Weingarten map depending on if the change of the underlying tangent vector is described in the original manifold or in Euclidean space (see, *e.g.*, chapter 3 in [27]).

² A critical point is degenerate if the Hessian H_θ at this point is singular (*i.e.*, $\det(H_\theta) = 0$). At degenerate critical points, one cannot use the eigenvalues of H_θ to determine if the critical point is a minimum (positive definite H_θ) or a maximum (negative definite H_θ). Geometrically, at a degenerate critical point, a quadratic approximation fails to capture the local behavior of the function that one wishes to study.

independent decreasing dimensions of L near θ^* (*i.e.*, the number of negative eigenvalues of the Hessian H_θ at that point).

In the standard basis, the Hessian is

$$H_\theta := \nabla_\theta \nabla_\theta L(\theta) = \begin{pmatrix} \frac{\partial^2 L}{\partial \theta_1^2} & \cdots & \frac{\partial^2 L}{\partial \theta_1 \partial \theta_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 L}{\partial \theta_N \partial \theta_1} & \cdots & \frac{\partial^2 L}{\partial \theta_N^2} \end{pmatrix}. \quad (3)$$

Closely related to the Hessian is the Fisher information matrix (FIM) [29, 30], whose elements F_{ij} are given by

$$F_{ij} = \mathbb{E} \left[\frac{\partial}{\partial \theta_i} \log p(X | \theta) \frac{\partial}{\partial \theta_j} \log p(X | \theta) \right], \quad (4)$$

where the expectation is taken over the random variable X and $p(x | \theta)$ denotes the probability density function associated with X conditioned on the parameter θ . The probability that X falls within the infinitesimal interval $[x, x + dx]$, given a known value of θ , is $p(x | \theta) dx$. Provided that certain regularity conditions such as

$$\frac{d}{d\theta_i} \mathbb{E} \left[\frac{\partial}{\partial \theta_j} \log p(X | \theta) \right] = \int \frac{\partial}{\partial \theta_i} \left[\left(\frac{\partial}{\partial \theta_j} \log p(x | \theta) \right) p(x | \theta) \right] dx \quad (5)$$

are satisfied [29, 31], we may rewrite Eq. (4) as

$$F_{ij} = -\mathbb{E} \left[\frac{\partial^2}{\partial \theta_i \partial \theta_j} \log p(X | \theta) \right]. \quad (6)$$

Notice the structural similarity between F_{ij} and the elements of H_θ [see Eq. (3)]. In natural gradient descent [32], the metric associated with a stochastic feedforward neural network is the FIM.

Considering a neural network with input-output pairs (x, y) and parameters θ , a possible choice of a model distribution, which takes on the role of the probability density in Eq. (4), is $p(x, y | \theta) = p(y | x, \theta) q(x)$ with $p(y | x, \theta) = \exp(-\|y - f_\theta(x)\|_2^2 / 2) / \sqrt{2\pi}$. Here, $q(x)$ denotes the distribution of inputs x , $f_\theta(x)$ is the output of a neural network given input x , and $\|\cdot\|_2$ is the Euclidean norm [33]. The expected values associated with FIM elements F_{ij} for $p(x, y | \theta)$ are calculated with respect to (w.r.t.) input-output pairs (x, y) of the joint distribution $p(x, y | \theta)$.

For a set of input samples $x(t)$ ($t \in \{1, \dots, T\}$) and corresponding outputs $f_\theta(t) \equiv f_\theta(x(t))$, the empirical Fisher information matrix is

$$F_\theta = \frac{1}{T} \sum_{t=1}^T \sum_{k=1}^C \nabla_\theta f_{\theta,k}(t) \nabla_\theta f_{\theta,k}(t)^\top, \quad (7)$$

where $f_{\theta,k}$ is the k -th entry of the neural network output and $k \in \{1, \dots, C\}$ [33, 34]. The empirical FIM converges towards the FIM, Eq. (4), as $T \rightarrow \infty$.

Given the mean squared loss function

$$L(\theta) = \frac{1}{2T} \sum_{t=1}^T \|y(t) - f_\theta(t)\|_2^2, \quad (8)$$

the Hessian H_θ of $L(\theta)$ and the empirical FIM F_θ are connected via

$$H_\theta = F_\theta - \frac{1}{T} \sum_{t=1}^T \sum_{k=1}^C (y_k(t) - f_{\theta,k}(t)) \nabla_\theta \nabla_\theta f_{\theta,k}(t). \quad (9)$$

Here, $y(t)$ denotes the desired output associated with sample $x(t)$. At the global optimum, which is usually attained for overparameterized neural networks [1], the difference $y_k(t) - f_{\theta,k}(t)$ vanishes for all k, t , and the empirical FIM is equal to the Hessian H_θ [34].

We can also establish a connection between H_θ and F_θ for the cross-entropy loss $L(\theta) = -1/T \sum_{t=1}^T \sum_{k=1}^C y_k(t) \log(f_{\theta,k}(t))$, where softmax outputs $f_{\theta,k}(t)$ are used to approximate binary class labels $y_k(t)$.³ Similarly to Eq. (9), we obtain

³ In this case, the underlying conditional probability distribution is $p(y|x, \theta) = \prod_{k=1}^C (f_{\theta,k}(t))^{y_k}$ [35].

$$H_\theta = \frac{1}{T} \sum_{t=1}^T \sum_{k=1}^C \frac{y_k(t)(\nabla_\theta f_{\theta,k}(t)\nabla_\theta f_{\theta,k}(t)^\top - f_{\theta,k}(t)\nabla_\theta \nabla_\theta f_{\theta,k}(t))}{f_{\theta,k}(t)^2}. \quad (10)$$

Given the scenario in which $f_{\theta,k}(t)$ approaches $y_k(t)$ for all k, t and using $\sum_{k=1}^C f_{\theta,k}(t) = 1$ (i.e., $\sum_{k=1}^C \nabla_\theta \nabla_\theta f_{\theta,k}(t) = 0$), we again have $H_\theta = F_\theta$ where

$$F_\theta = \frac{1}{T} \sum_{t=1}^T \sum_{k=1}^C \frac{1}{f_{\theta,k}(t)} \nabla_\theta f_{\theta,k}(t)\nabla_\theta f_{\theta,k}(t)^\top \quad (11)$$

is the empirical FIM associated with the cross-entropy loss [35].

The neural network examples that we consider in Section IV are trained using cross-entropy loss. In the Appendix, we also consider a function approximation problem in which we train neural networks using a mean squared loss function.

B. Random projections

To graphically explore an N -dimensional loss function L around a critical point θ^* , one may wish to work in a lower-dimensional representation. For example, a two-dimensional projection of L around θ^* is provided by

$$L(\theta^* + \alpha\eta + \beta\delta), \quad (12)$$

where the parameters $\alpha, \beta \in \mathbb{R}$ scale the directions $\eta, \delta \in \mathbb{R}^N$. The corresponding graph representation is $(\alpha, \beta, L(\alpha, \beta)) \subseteq \mathbb{R}^3$.

In high-dimensional spaces, there exist vastly many more almost-orthogonal than orthogonal directions. In fact, if the dimension of our space is large enough, with high probability, random vectors will be sufficiently close to orthogonal [36]. Following this result, many related works [13, 15, 16] use random Gaussian directions with independent and identically distributed vector elements $\eta_i, \delta_i \sim \mathcal{N}(0, 1)$ ($i \in \{1, \dots, N\}$).

We now turn to showing that η, δ are almost orthogonal using a concentration inequality for chi-squared distributed random variables. To do so, we first note that the scalar product of random Gaussian vectors η, δ is a sum of the difference between two chi-squared distributed random variables because

$$\sum_{i=1}^N \eta_i \delta_i = \sum_{i=1}^N \frac{1}{4} (\eta_i + \delta_i)^2 - \frac{1}{4} (\eta_i - \delta_i)^2 = \sum_{i=1}^N \frac{1}{2} X_i^2 - \frac{1}{2} Y_i^2, \quad (13)$$

where $X_i, Y_i \sim \mathcal{N}(0, 1)$. In the last step of Eq. (13), we rescaled X_i and Y_i by a factor $\sqrt{2}$ to obtain normally distributed random variables with variance 2. Since $\mathbb{E}[Z] = N$ for $Z = \sum_{i=1}^N X_i^2$, the right-hand side of Eq. (13) vanishes in expectation. That is,

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N \eta_i \delta_i = 0. \quad (14)$$

Hence, the vectors η, δ are orthogonal in the limit $N \rightarrow \infty$. For finite N , we can bound Eq. (13) using the concentration inequalities

$$\Pr \left(\frac{1}{N} \sum_{i=1}^N X_i^2 - 1 \geq \epsilon + \frac{1}{2}\epsilon^2 \right) \leq e^{-\frac{N\epsilon^2}{4}} \quad (15)$$

$$\Pr \left(1 - \frac{1}{N} \sum_{i=1}^N X_i^2 \geq \epsilon \right) \leq e^{-\frac{N\epsilon^2}{4}} \quad (16)$$

for any $\epsilon > 0$ [37]. In the limit $\epsilon \rightarrow 0$, the above inequalities can be cast in the following form

$$\Pr \left(\left| \frac{1}{N} \sum_{i=1}^N X_i^2 - 1 \right| \geq \epsilon \right) \leq 2e^{-\frac{N\epsilon^2}{4}}. \quad (17)$$

Symbol	Definition
$H_\theta \in \mathbb{R}^{N \times N}$	Hessian in the original loss space
$\kappa_i^\theta \in \mathbb{R}$	principal curvatures in the original loss space, where $i \in \{1, \dots, N\}$ (<i>i.e.</i> , the eigenvalues of H_θ)
$H_{\alpha,\beta} \in \mathbb{R}^{2 \times 2}$	Hessian in a two-dimensional projection of an N -dimensional loss function
$\kappa_\pm^{\alpha,\beta} \in \mathbb{R}$	principal curvatures in a two-dimensional loss projection (<i>i.e.</i> , the eigenvalues of $H_{\alpha,\beta}$)
$\bar{\kappa}^{\alpha,\beta} \in \mathbb{R}$	principal curvature in the expected, two-dimensional loss projection (<i>i.e.</i> , the eigenvalue of $\mathbb{E}[H_{\alpha,\beta}]$)
$H \in \mathbb{R}$	mean curvature (<i>i.e.</i> , $\sum_{i=1}^N \kappa_i^\theta / N = \bar{\kappa}^{\alpha,\beta} / N$)

TABLE I. **Overview of several quantities that are useful to characterize curvature of high-dimensional loss functions.** We summarize the employed definitions of Hessians and several derived curvature measures that are useful to quantify curvature of N -dimensional loss functions $L(\theta^*): \mathbb{R}^N \rightarrow \mathbb{R}$ and their two-dimensional random projections $L(\theta^* + \alpha\eta + \beta\delta)$ at a non-degenerate critical point θ^* . The vectors $\eta, \delta \in \mathbb{R}^N$ are the directions along which we project the original loss function and $\alpha, \beta \in \mathbb{R}$ are the corresponding scale factors. Random loss projections are usually based on Gaussian vectors η, δ , which are almost orthogonal in high-dimensional spaces (*i.e.*, for large N).

Using $|N^{-1} \sum_i \eta_i \delta_i| = |(2N)^{-1} \sum_i X_i^2 - 1 + 1 - Y_i^2|$, we obtain

$$\Pr \left(\left| \frac{1}{N} \sum_{i=1}^N \eta_i \delta_i \right| \geq \epsilon \right) \leq \sqrt{2} e^{-\frac{N\epsilon^2}{2}} \quad (18)$$

in the limit of large N . For further details on the derivation of Eq. (18), see Appendix A.

C. Principal curvature

With the form of random Gaussian projections in hand, we now analyze the principal curvatures in both the original and lower-dimensional spaces. The Hessian associated with the two-dimensional loss projection (12) is

$$\begin{aligned} H_{\alpha,\beta} &= \begin{pmatrix} \frac{\partial^2 L}{\partial \alpha^2} & \frac{\partial^2 L}{\partial \alpha \partial \beta} \\ \frac{\partial^2 L}{\partial \beta \partial \alpha} & \frac{\partial^2 L}{\partial \beta^2} \end{pmatrix} = \begin{pmatrix} \sum_{i,j} \eta_i \eta_j \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} & \sum_{i,j} \eta_i \delta_j \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} \\ \sum_{i,j} \eta_i \delta_j \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} & \sum_{i,j} \delta_i \delta_j \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} \end{pmatrix} \\ &= \begin{pmatrix} (H_\theta)_{ij} \eta^i \eta^j & (H_\theta)_{ij} \eta^i \delta^j \\ (H_\theta)_{ij} \eta^i \delta^j & (H_\theta)_{ij} \delta^i \delta^j \end{pmatrix}, \end{aligned} \quad (19)$$

where we use Einstein notation in the last equality.

In analogy with Eq. (9), the dimension-reduced Hessian $H_{\alpha,\beta}$ is connected to the corresponding FIM $F_{\alpha,\beta}$ according to

$$H_{\alpha,\beta} = F_{\alpha,\beta} - \frac{1}{T} \sum_{t=1}^T \sum_{k=1}^C (y_k(t) - f_{(\alpha,\beta),k}(t)) \nabla_{(\alpha,\beta)} \nabla_{(\alpha,\beta)} f_{(\alpha,\beta),k}(t), \quad (20)$$

where $f_{(\alpha,\beta),k}(t)$ is the k -th entry of the output of a neural network with parameters $\theta^* + \alpha\eta + \beta\delta$. The operator $\nabla_{(\alpha,\beta)}$ denotes the gradient w.r.t. parameters α, β . Recall that Eq. (9) [and hence Eq. (20)] is based on a mean squared loss function. One may also use Eq. (10) to obtain a connection between the dimension-reduced Hessian and the corresponding cross-entropy FIM.

Lower-dimensional approximations $H_{\alpha,\beta}$ of the Hessian H_θ in the original high-dimensional space might be useful in generalized Gauss–Newton algorithms that are based on positive semidefinite and computationally affordable Hessian approximations [38, 39].

Because the elements of δ, η arising in $H_{\alpha,\beta}$ are distributed according to a standard normal distribution, the second derivatives of the loss function L in Eq. (19) have prefactors that are products of standard normal variables and,

hence, can be expressed as sums of chi-squared distributed random variables as in Eq. (13). To summarize, elements of $H_{\alpha,\beta}$ are sums of second derivatives of L in the original space weighted with chi-squared distributed prefactors.

The principal curvatures $\kappa_{\pm}^{\alpha,\beta}$ (*i.e.*, the eigenvalues of $H_{\alpha,\beta}$) are

$$\kappa_{\pm}^{\alpha,\beta} = \frac{1}{2} \left(A + C \pm \sqrt{4B^2 + (A - C)^2} \right), \quad (21)$$

where $A = (H_{\theta})_{ij}\eta^i\eta^j$, $B = (H_{\theta})_{ij}\eta^i\delta^j$, and $C = (H_{\theta})_{ij}\delta^i\delta^j$. To the best of our knowledge, a closed, analytic expression for the distribution of the quantities A, B, C is not yet known [37, 40–42].

We now turn to the relationship between random projections and principal curvature. The appeal of random projections is that pairwise distances between points in a high-dimensional space can be nearly preserved by a lower-dimensional linear embedding, affording a low-dimensional representation of mean and variance information with minimal distortion [43]. The question is whether random projections also preserve curvature information, and if so, what is the nature of the relationship between random Gaussian directions and principal curvature. For instance, Li *et al.* [13] assert that “the principal curvatures of a dimensionality-reduced plot (with random Gaussian directions) are weighted averages of the principal curvatures of the full-dimensional surface”. However, our results show that the principal curvatures $\kappa_{\pm}^{\alpha,\beta}$ in a two-dimensional loss projection are weighted averages of the Hessian elements $(H_{\theta})_{ij}$ in the original space and not weighted averages of the principal curvatures κ_i^{θ} , which instead are solutions of an N -th degree polynomial. Similar arguments apply to projections with dimension larger than 2.

In Section III we discuss examples that show that lower-dimensional projections of $L(\theta)$ can be misleading, since high-dimensional saddle points may appear to be minima, maxima, or almost flat regions depending on the index of the underlying Hessian H_{θ} in the original space.

Returning to principal curvature, since $\sum_{i,j} a_{ij}\eta^i\eta^j = \sum_i a_{ii}\eta^i\eta^i + \sum_{i \neq j} a_{ij}\eta^i\eta^j$ ($a_{ij} \in \mathbb{R}$), we find that $\mathbb{E}[A] = \mathbb{E}[C] = (H_{\theta})_{ii}^i$ and $\mathbb{E}[B] = 0$ where $(H_{\theta})_{ii}^i \equiv \text{tr}(H_{\theta}) = \sum_{i=1}^N \kappa_i^{\theta}$.⁴ To show that the expected values of $a_{ij}\eta^i\eta^j$ ($i \neq j$) or $a_{ij}\eta^i\delta^j$ vanish, one can either invoke independence of η^i, η^j ($i \neq j$) and η^i, δ^j or transform both products into corresponding differences of two chi-squared random variables with the same mean [see Eq. (13)].

Hence, the expected, dimension-reduced Hessian (19) is

$$\mathbb{E}[H_{\alpha,\beta}] = \begin{pmatrix} (H_{\theta})_{ii}^i & 0 \\ 0 & (H_{\theta})_{ii}^i \end{pmatrix}. \quad (22)$$

The corresponding eigenvalue (or principal curvature) $\bar{\kappa}^{\alpha,\beta}$ is therefore given by the sum over all principal curvatures in the original space (*i.e.*, $\bar{\kappa}^{\alpha,\beta} = \sum_{i=1}^N \kappa_i^{\theta}$). Hence, the value of the principal curvature $\bar{\kappa}^{\alpha,\beta}$ in the expected dimension-reduced space will be either positive (if the positive principal curvatures in the original space dominate), negative (if the negative principal curvatures in the original space dominate), or close to zero (if positive and negative principal curvatures in the original space cancel out each other). As a result, saddle points will not appear as such in the expected random projection.

In addition to the connection between $\bar{\kappa}^{\alpha,\beta}$ and the principal curvatures κ_i^{θ} , we now provide an overview of additional mathematical relations between different curvature measures that are useful to quantify curvature properties of high-dimensional loss functions and their two-dimensional random projections.

Invoking Eq. (21), we can relate $\bar{\kappa}^{\alpha,\beta}$ to $\text{tr}(H_{\theta})$ and $\kappa_{\pm}^{\alpha,\beta}$. Because $\kappa_{+}^{\alpha,\beta} + \kappa_{-}^{\alpha,\beta} = A + C$, we have

$$\text{tr}(H_{\theta}) = \bar{\kappa}^{\alpha,\beta} = \sum_{i=1}^N \kappa_i^{\theta} = \frac{1}{2} \left(\mathbb{E}[\kappa_{+}^{\alpha,\beta}] + \mathbb{E}[\kappa_{-}^{\alpha,\beta}] \right). \quad (23)$$

The mean curvature H in the original space is related to $\bar{\kappa}^{\alpha,\beta}$ via

$$H = \frac{1}{N} \bar{\kappa}^{\alpha,\beta} = \frac{1}{N} \sum_{i=1}^N \kappa_i^{\theta}. \quad (24)$$

We summarize the definitions of the employed Hessians and curvature measures in Tab. I.

⁴ The expected values of the quantities A, B , and C correspond to ensemble means (29) over S realizations of random projections in the limit $S \rightarrow \infty$.

D. Hessian trace estimates

Finally, we point to a connection between the above curvature measures and existing Hessian trace estimates. The trace (or unnormalized mean curvature) of the Hessian H_θ has already found applications in characterizing loss landscapes of neural networks [21, 44]. A common way of estimating $\text{tr}(H_\theta)$ without explicitly computing all eigenvalues of H_θ is based on Hutchinson's method [19] and random numerical linear algebra [45, 46]. The basic idea behind this approach is to (i) use a random vector $z \in \mathbb{R}^N$ with elements z_i that are distributed according to a distribution function with zero mean and unit variance (e.g., a Rademacher distribution with $\Pr(z_i = \pm 1) = 1/2$), and (ii) compute $z^\top H_\theta z$, an unbiased estimator of $\text{tr}(H_\theta)$, using Hessian-vector products. That is,

$$\text{tr}(H_\theta) = \mathbb{E}[z^\top H_\theta z]. \quad (25)$$

Recall that Eq. (23) shows that the principal curvature of the expected random loss projection, $\bar{\kappa}^{\alpha,\beta}$, is equal to $\text{tr}(H_\theta)$. Instead of estimating $\text{tr}(H_\theta)$ using the original Hutchinson's method (25), an alternative Hutchinson-type estimate is provided by the mean of the expected values of $\kappa_-^{\alpha,\beta}$ and $\kappa_+^{\alpha,\beta}$ [see Eq. (23)].

III. ILLUSTRATIVE EXAMPLES

In this section, we summarize the main concepts derived in Section II in terms of different examples. These examples are intended to aid intuition on how (i) principal and mean curvatures of the original loss function relate to those of random projections, (ii) curvature-based Hessian trace estimates relate to those obtained with the original Hutchinson's method, and (iii) Hessian directions (i.e., the eigenbasis of the Hessian H_θ of $L(\theta)$) can be used for a guided visual analysis of saddle information in high-dimensional spaces.

A. Extracting curvature information

We now study two examples that will help build intuition for the concepts discussed in the previous sections. In the first example, we study a critical point θ^* of an N -dimensional loss function $L(\theta)$ for which (i) all principal curvatures have the same magnitude and (ii) the number of positive curvature directions is equal to the number of negative curvature directions. The mean curvature of this saddle point is zero. In accordance with Eq. (24), we will show that the mean curvature can be approximated by a function of ensemble means of elements of the dimension-reduced Hessian $H_{\alpha,\beta}$, and that the principal curvatures $\kappa_\pm^{\alpha,\beta}$ are able to correctly identify the saddle point in the majority of simulated projections. In the second example, we use a loss function associated with an unequal number of negative and positive curvature directions. For the different curvature measures derived in Section II C, we find that random projections cannot identify the underlying saddle point.

The loss function of our first example is

$$L(\theta) = \frac{1}{2}\theta_{2n+1} \left(\sum_{i=1}^n \theta_i^2 - \theta_{i+n}^2 \right), \quad n \in \mathbb{Z}_+, \quad (26)$$

where we set $N = 2n + 1$. A critical point θ^* of the loss function (26) satisfies

$$(\nabla_\theta L)(\theta^*) = \begin{pmatrix} \theta_1^* \theta_{2n+1}^* \\ \vdots \\ \theta_n^* \theta_{2n+1}^* \\ -\theta_{n+1}^* \theta_{2n+1}^* \\ \vdots \\ -\theta_{2n}^* \theta_{2n+1}^* \\ \frac{1}{2} \left(\sum_{i=1}^n \theta_i^{*2} - \theta_{i+n}^{*2} \right) \end{pmatrix} = 0. \quad (27)$$

The Hessian at the critical point $\theta^* = (\theta_1^*, \dots, \theta_{2n}^*, \theta_{2n+1}^*) = (0, \dots, 0, 1)$ is

$$H_\theta = \text{diag}(\underbrace{1, \dots, 1}_{n \text{ times}}, \underbrace{-1, \dots, -1}_{n \text{ times}}, 0). \quad (28)$$

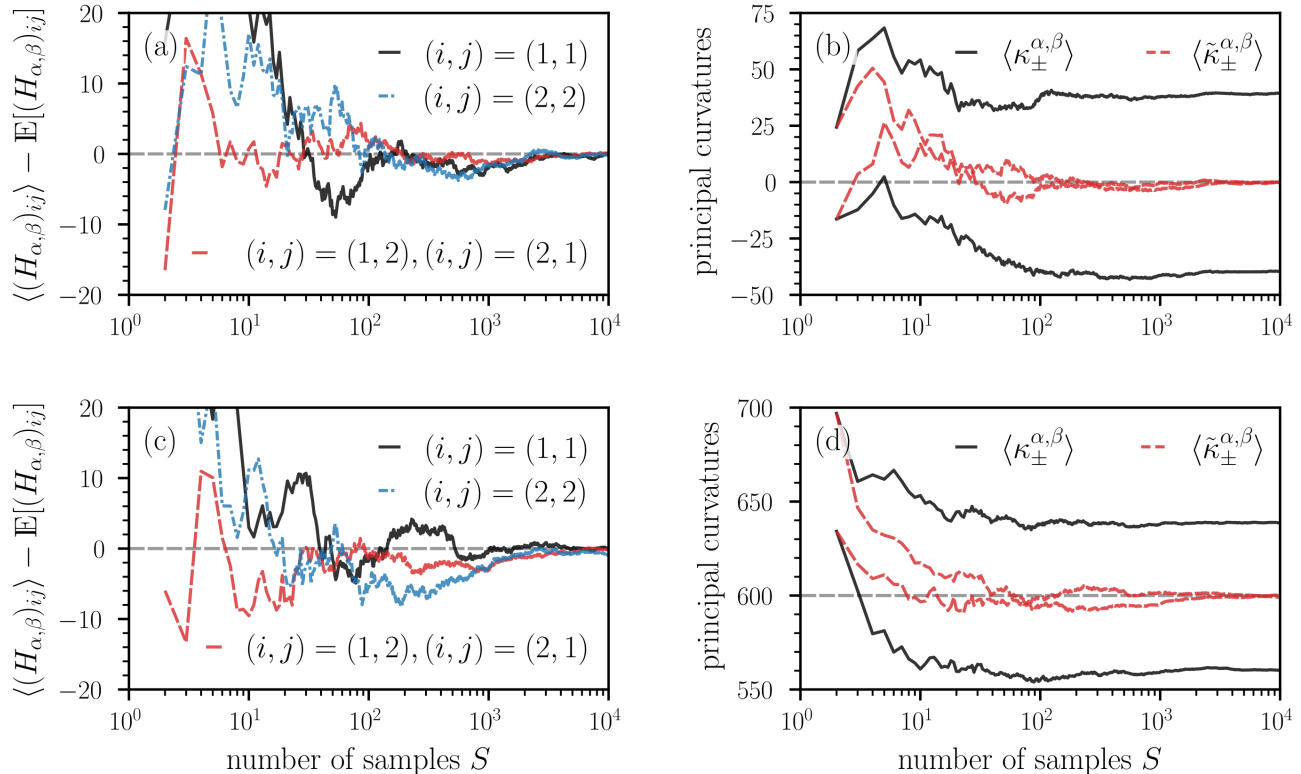


FIG. 1. Convergence of the ensemble mean (29) of Hessian elements and curvatures measures as a function of the number of random projections S . (a,c) The deviation of the ensemble means $\langle (H_{\alpha,\beta})_{ij} \rangle$ ($i, j \in \{1, 2\}$) of Hessian elements from the corresponding expected values as a function of S . Notice that the expected value of the diagonal elements $(H_{\alpha,\beta})_{11}$ and $(H_{\alpha,\beta})_{22}$ is equal to $\bar{\kappa}^{\alpha,\beta}$ (i.e., to the sum of principal curvatures in the original space) [see Eqs. (22) and (23)]. A relatively large number of random projections between 10^3 and 10^4 is required to keep the deviations at values smaller than about 2–4. (b,d) The ensemble means $\langle \kappa_{\pm}^{\alpha,\beta} \rangle$ [see Eq. (21)] and $\langle \tilde{\kappa}_{\pm}^{\alpha,\beta} \rangle$ [see Eq. (30)] as a function of S . Dashed grey lines represent $\bar{\kappa}^{\alpha,\beta} = \text{tr}(H_{\theta})$. In panels (a,b) and (c,d), the N -dimensional loss functions are given by Eqs. (26) and (31), respectively. We evaluate the corresponding Hessians (28) and (32) at the saddle point $\theta^* = (\theta_1^*, \dots, \theta_{2n}^*, \theta_{2n+1}^*) = (0, \dots, 0, 1)$. In both loss functions, we set $n = 500$ and in loss function (31) we set $\tilde{n} = 800$.

Because H_{θ} has positive and negative eigenvalues, the critical point is a saddle. The corresponding principal curvatures are $\kappa_i^{\theta} \in \{-1, 1\}$ ($i \in \{1, \dots, N-1\}$) and $\kappa_N^{\theta} = 0$. In this example, the mean curvature H , as defined in Eq. (24), is equal to 0. According to Eq. (22), the principal curvature, $\bar{\kappa}^{\alpha,\beta}$, associated with the expected, dimension-reduced Hessian $H_{\alpha,\beta}$ is also equal to 0, erroneously indicating an apparently flat loss landscape if one would use $\bar{\kappa}^{\alpha,\beta}$ as the main measure of curvature. To compare the convergence of different curvature measures as a function of the number of loss projections S , we will now study the ensemble mean

$$\langle X \rangle = \frac{1}{S} \sum_{k=1}^S X^{(k)} \quad (29)$$

of different quantities of interest X such as Hessian elements and principal curvature measures in dimension-reduced space. Here, $X^{(k)}$ is the k -th realization (or sample) of X .

We first study the dependence of ensemble means $\langle (H_{\alpha,\beta})_{ij} \rangle$ ($i, j \in \{1, 2\}$) of elements of the dimension-reduced Hessian $H_{\alpha,\beta}$ on the number of samples S . According to Eq. (22), the diagonal elements of $\mathbb{E}[H_{\alpha,\beta}]$ are proportional to the mean curvature of the original high-dimensional space and are thus useful to examine curvature properties of high-dimensional loss functions. Figure 1(a) shows the convergence of the ensemble means $\langle (H_{\alpha,\beta})_{ij} \rangle$ towards the expected values $\mathbb{E}[(H_{\alpha,\beta})_{ij}]$ as a function of S . Note that $\mathbb{E}[(H_{\alpha,\beta})_{ij}] = 0$ for all i, j . For a few dozen loss projections, the deviations of the ensemble means from the corresponding expected values reach values larger than 20. A relatively large number of loss projections S between 10^3 – 10^4 is required to keep these deviations at values that are smaller than about 2–4. The solid black and red lines in Figure 1(b), respectively, show the ensemble means $\langle \kappa_{\pm}^{\alpha,\beta} \rangle$ and the

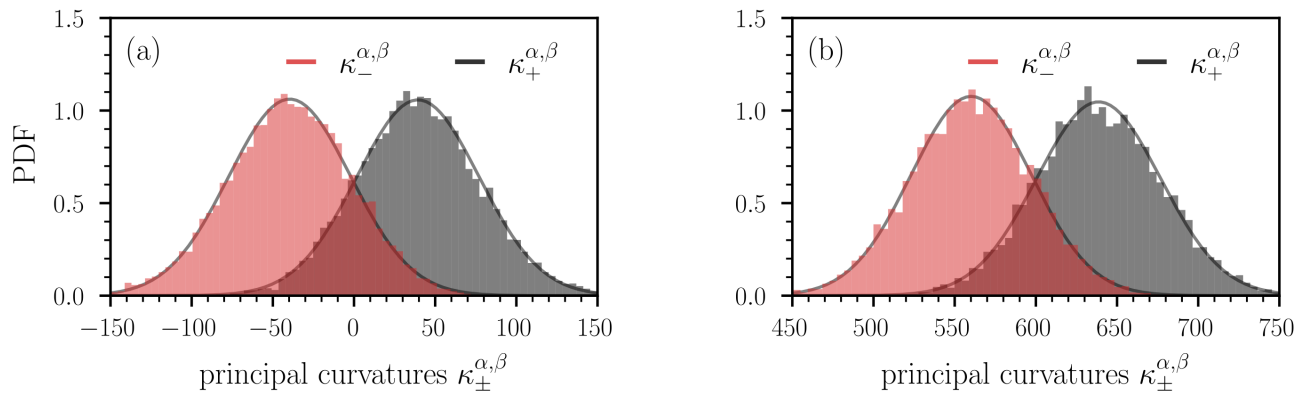


FIG. 2. Distribution of principal curvatures $\kappa_-^{\alpha,\beta}$ (red bars) and $\kappa_+^{\alpha,\beta}$ (black bars). In panels (a) and (b), the loss functions are given by Eqs. (26) and (31), respectively. We evaluate the corresponding Hessians (28) and (32) at the saddle point $\theta^* = (\theta_1^*, \dots, \theta_{2n}^*, \theta_{2n+1}^*) = (0, \dots, 0, 1)$. In both loss functions, we set $n = 500$ and in loss function (31) we set $\tilde{n} = 800$. While in panel (a), the probability $\Pr(\kappa_+^{\alpha,\beta} \kappa_-^{\alpha,\beta} > 0)$ that the critical point in the lower-dimensional, random projection does not appear as a saddle is about 0.3, it is 1 in panel (b). Histograms are based on 10,000 random projections that are used to compute $\kappa_{\pm}^{\alpha,\beta}$. Solid grey lines indicate Gaussian approximations of the empirical distributions.

eigenvalues

$$\langle \tilde{\kappa}_{\pm}^{\alpha,\beta} \rangle = \frac{1}{2} \left(\langle A \rangle + \langle C \rangle \pm \sqrt{4\langle B \rangle^2 + (\langle A \rangle - \langle C \rangle)^2} \right) \quad (30)$$

of the ensemble-averaged dimension-reduced Hessian as a function of S . Since $\mathbb{E}[A] = \mathbb{E}[C] = (H_{\theta})^i_i$ and $\mathbb{E}[B] = 0$ [see Eq. (22)], we have that $\langle \tilde{\kappa}_{\pm}^{\alpha,\beta} \rangle = \bar{\kappa}^{\alpha,\beta}$ in the limit $S \rightarrow \infty$. In the current example, the ensemble means $\langle \tilde{\kappa}_{\pm}^{\alpha,\beta} \rangle$ thus approach $\bar{\kappa}^{\alpha,\beta} = 0$ for large numbers of samples S , represented by the dashed red lines in Figure 1(b). The ensemble means $\langle \kappa_{\pm}^{\alpha,\beta} \rangle$ converge towards values of opposite sign, indicating a saddle point.

For a sample size of $S = 10^4$, we show the distribution of the principal curvatures $\kappa_{\pm}^{\alpha,\beta}$ in Figure 2(a). We observe that the distributions are plausibly Gaussian. We also calculate the probability $\Pr(\kappa_+^{\alpha,\beta} \kappa_-^{\alpha,\beta} > 0)$ that the critical point in the lower-dimensional, random projection does not appear as a saddle (*i.e.*, $\kappa_+^{\alpha,\beta} \kappa_-^{\alpha,\beta} > 0$). For the example shown in Figure 2(a), we find that $\Pr(\kappa_+^{\alpha,\beta} \kappa_-^{\alpha,\beta} > 0) \approx 0.3$. That is, in about 30% of the simulated projections, the lower-dimensional loss landscape wrongly indicates that it does not correspond to a saddle.

Our first example, which is based on the loss function (26), shows that the principal curvatures in the lower-dimensional representation of $L(\theta)$ may capture the saddle behavior in the original space if one computes ensemble means $\langle \kappa_{\pm}^{\alpha,\beta} \rangle$ in the lower-dimensional space [Figure 1(b)]. However, if one first calculates ensemble means of the elements of the dimension-reduced Hessian $H_{\alpha,\beta}$ to infer $\langle \tilde{\kappa}_{\pm}^{\alpha,\beta} \rangle$, the loss landscape appears to be flat in this example. We thus conclude that different ways of computing ensemble means (either before or after calculating the principal curvatures) may lead to different results with respect to the “flatness” of a dimension-reduced loss landscape.

In the next example, we will show that random projections cannot identify certain saddle points regardless of the underlying averaging process. We consider the loss function

$$L(\theta) = \frac{1}{2} \theta_{2n+1} \left(\sum_{i=1}^{\tilde{n}} \theta_i^2 - \sum_{i=\tilde{n}+1}^{2n} \theta_i^2 \right), \quad n \in \mathbb{Z}_+, n < \tilde{n} \leq 2n, \quad (31)$$

where we use the convention $\sum_{i=a}^b (\cdot) = 0$ if $a > b$. The Hessian at the critical point $(\theta_1^*, \dots, \theta_{2n}^*, \theta_{2n+1}^*) = (0, \dots, 0, 1)$ is

$$H_{\theta} = \text{diag}(\underbrace{1, \dots, 1}_{\tilde{n} \text{ times}}, \underbrace{-1, \dots, -1}_{2n - \tilde{n} \text{ times}}, 0). \quad (32)$$

As in the previous example, the critical point is again a saddle, but the mean curvature is $H = 2(\tilde{n} - n)/N > 0$. In the following numerical experiments, we set $n = 500$ and $\tilde{n} = 800$. Figure 1(c) shows that the ensemble means

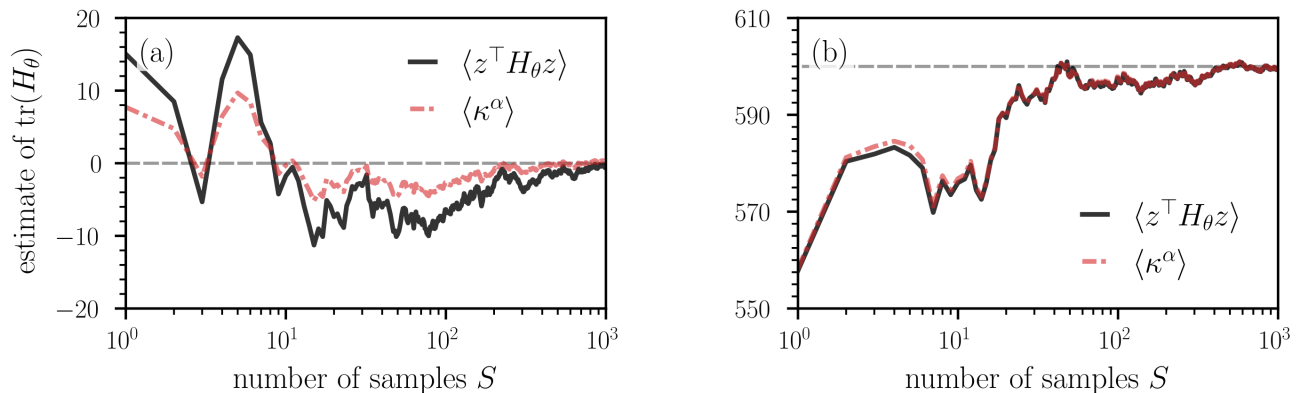


FIG. 3. Estimating the trace of the Hessian H_θ . In panels (a) and (b), the loss functions are given by Eqs. (26) and (31), respectively. We evaluate the corresponding Hessians (28) and (32) at the saddle point $\theta^* = (\theta_1^*, \dots, \theta_{2n}^*, \theta_{2n+1}^*) = (0, \dots, 0, 1)$. In both loss functions, we set $n = 500$ and in loss function (31) we set $\tilde{n} = 800$. Solid black and dash-dotted red lines represent Hutchinson [$\langle z^\top H_\theta z \rangle$; see Eq. (25)] and curvature-based ($\langle \kappa^\alpha \rangle$) estimates of $\text{tr}(H_\theta)$, respectively. We compute ensemble means $\langle \cdot \rangle$ as defined in Eq. (29) for different numbers of random projections S . The trace estimates in panels (a) and (b), respectively, converge towards the true trace values $\text{tr}(H_\theta) = 0$ and $\text{tr}(H_\theta) = 600$ that are indicated by dashed grey lines. In both methods, the same random vectors with elements that are distributed according to a standard normal distribution are used. For the curvature-based estimation of $\text{tr}(H_\theta)$, we perform least-square fits of $L(\theta^* + \alpha\eta)$ over an interval $\alpha \in [-0.05, 0.05]$.

$\langle (H_{\alpha,\beta})_{ij} \rangle$ converge towards the expected value $\mathbb{E}[(H_{\alpha,\beta})_{ij}]$ as the number of samples increases. We again observe that a relatively large number of random loss projections S between 10^3 and 10^4 is required to keep the deviations of ensemble means from their corresponding expected values small. Because of the dominance of positive principal curvatures κ_i^θ in the original space, the corresponding ensemble means of principal curvatures (*i.e.*, $\langle \kappa_\pm^{\alpha,\beta} \rangle$, $\langle \tilde{\kappa}_\pm^{\alpha,\beta} \rangle$) in the lower-dimensional representation approach positive values [Figure 1(d)]. The distribution of $\kappa_\pm^{\alpha,\beta}$ indicates that the probability of observing a saddle in the lower-dimensional loss landscape is vanishingly small [Figure 2(b)]. In this second example, both ways of computing ensemble means, before and after calculating the lower-dimensional principal curvatures, mistakenly suggest that the saddle in the original space is a minimum in dimension-reduced space.

To summarize, for both loss functions (26) and (31), the saddle point $\theta^* = (0, \dots, 0, 1)$ in the original loss function $L(\theta)$ is often misrepresented in lower-dimensional representations $L(\theta + \alpha\eta + \beta\delta)$ if random directions are used. Depending on (i) the employed curvature measure and (ii) the index of the underlying Hessian H_θ in the original space, the saddle $\theta^* = (0, \dots, 0, 1)$ often appears erroneously as either a minimum, maximum, or an almost flat region.

If the critical point were a minimum or maximum (*i.e.*, a critical point associated with a positive definite or negative definite Hessian H_θ), it would be correctly represented in the corresponding expected random projection because the sign of its principal curvature $\bar{\kappa}^{\alpha,\beta}$, which is proportional to the sum of all eigenvalues κ_i^θ of the Hessian H_θ in the original loss space, would be equal to the sign of the principal curvatures κ_i^θ . However, such points are scarce in high-dimensional loss spaces [17, 18].

Finally, because of the confounding factors associated with the inability of random projections to correctly identify saddle information, it does not appear advisable to use “flatness” around a critical point in a lower-dimensional random loss projection as a measure of generalization error [13].

B. Hessian trace

We now use the loss functions (26) and (31) to compare the convergence behavior between the original Hutchinson method (25) and the curvature-based trace estimation (23). Instead of two random directions, we use one random Gaussian direction η and perform quadratic least-square fits for 50 equidistant values of α in the interval $[-0.05, 0.05]$ to extract estimates of $\text{tr}(H_\theta)$ from $L(\theta^* + \alpha\eta)$. We use the same random Gaussian directions in Hutchinson’s method.

Figure 3 shows how the Hutchinson and curvature-based trace estimates converge towards the true trace values, 0 for the loss function (26) and 600 for the loss function (31) with $n = 500$ and $\tilde{n} = 800$. Given that we use the same random vectors in both methods, their convergence behavior towards the true trace value is similar. With the curvature-based method, one can produce Hutchinson-type trace estimates without computing Hessian-vector products. However, it requires the user to specify an appropriate interval for the quantity α so that the mean curvature can be properly

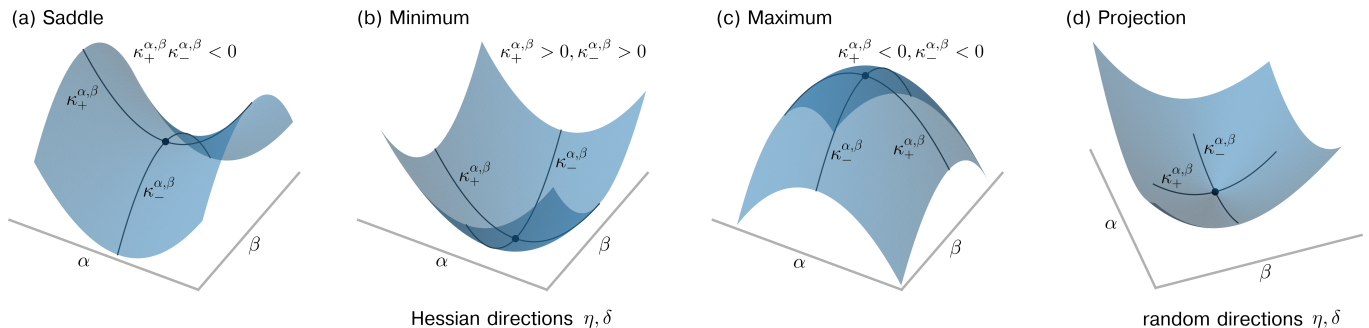


FIG. 4. Dimensionality-reduced loss $L(\theta^* + \alpha\eta + \beta\delta)$ of Eq. (31) with $n = 900, \tilde{n} = 1000$ for different directions η, δ . (a–c) The directions η, δ correspond to eigenvectors of the Hessian H_θ of Eq. (31). If the eigenvalues associated with η, δ have different signs, the corresponding loss landscape is a saddle as depicted in panel (a). If the eigenvalues associated with η, δ have the same sign, the corresponding loss landscape is either a minimum (both signs are positive) as shown in panel (b) or a maximum (both signs are negative) as shown in panel (c). Because there is an excess of $\tilde{n} - n = 100$ positive eigenvalues in H_θ , a projection onto a dimension-reduced space that is spanned by the random directions η, δ is often associated with an apparently convex loss landscape. An example of such an apparent minimum is shown in panel (d). We selected a single pair of random directions η, δ (*i.e.*, no averaging over random directions has been performed).

estimated in a quadratic-approximation regime. It also requires a sufficiently large number of points in this interval. Therefore, it may be less accurate than the original Hutchinson method.

C. Hessian directions

Given the described shortcomings of random projections (*e.g.*, in correctly identifying saddles of high-dimensional loss functions), we suggest to use Hessian directions (*i.e.*, the eigenbasis of H_θ) as directions η, δ in $L(\theta^* + \alpha\eta + \beta\delta)$. For Eq. (31) with $n = 900, \tilde{n} = 1000$, we show projections along different Hessian directions in Figure 4(a–c). We observe that different Hessian directions indicate different types of critical points in dimension-reduced space. If the eigenvalues associated with the Hessian directions η, δ have different signs, the corresponding lower-dimensional loss landscape is a saddle [Figure 4(a)]. If both eigenvalues have the same sign, the loss landscape is either a minimum [Figure 4(b): both signs are positive] or it is a maximum [Figure 4(c): both signs are negative]. If one uses a random projection instead, the resulting lower-dimensional loss landscape often appears to be a minimum in this example [Figure 4(d)]. To quantify the proportion of random projections that correctly identify the saddle with $n = 900, \tilde{n} = 1000$, we generated 10,000 realizations of $\kappa_\pm^{\alpha, \beta}$ [see Eq. (21)]. We find that the signs of $\kappa_\pm^{\alpha, \beta}$ were different in only about 0.5% of all simulated realizations. That is, in this example the principal curvatures $\kappa_\pm^{\alpha, \beta}$ indicate a saddle in only about 0.5% of the studied projections.

In the next section, we will show that Hessian directions that are associated with the largest-magnitude positive and negative eigenvalues of H_θ are useful to appropriately identify saddle information and visually study the geometric properties of saddle points in high-dimensional loss spaces of image classifiers.

IV. APPLICATIONS TO NEURAL NETWORKS

We now compare projections $L(\theta^* + \alpha\eta + \beta\delta)$ using (i) *dominant Hessian directions* (*i.e.*, eigenvectors that are associated with the largest-magnitude positive and negative eigenvalues of H_θ) and (ii) random directions. Hessian directions were computed using HVPs without an explicit representation of H_θ (see Algorithm 1). We first compute the largest-magnitude eigenvalue and then use an annihilation method [47] to compute the second largest-magnitude eigenvalue of opposite sign. More details on the annihilation algorithm are provided in Appendix B. Other deflation techniques can be used to find additional Hessian directions.

Instead of explicitly representing the Hessian H_θ , Algorithm 1 evaluates products between H_θ and a vector $v \in \mathbb{R}^N$ using the identity

$$\nabla_\theta [(\nabla_\theta L)^\top v] = (\nabla_\theta \nabla_\theta L)v + (\nabla_\theta L)^\top \nabla_\theta v = H_\theta v. \quad (33)$$

In the first step, the gradient $(\nabla_\theta L)^\top$ is computed using reverse-mode autodifferentiation (AD) to then compute the scalar product $[(\nabla_\theta L)^\top v]$. In the second step, we again apply reverse-mode AD to the computational graph

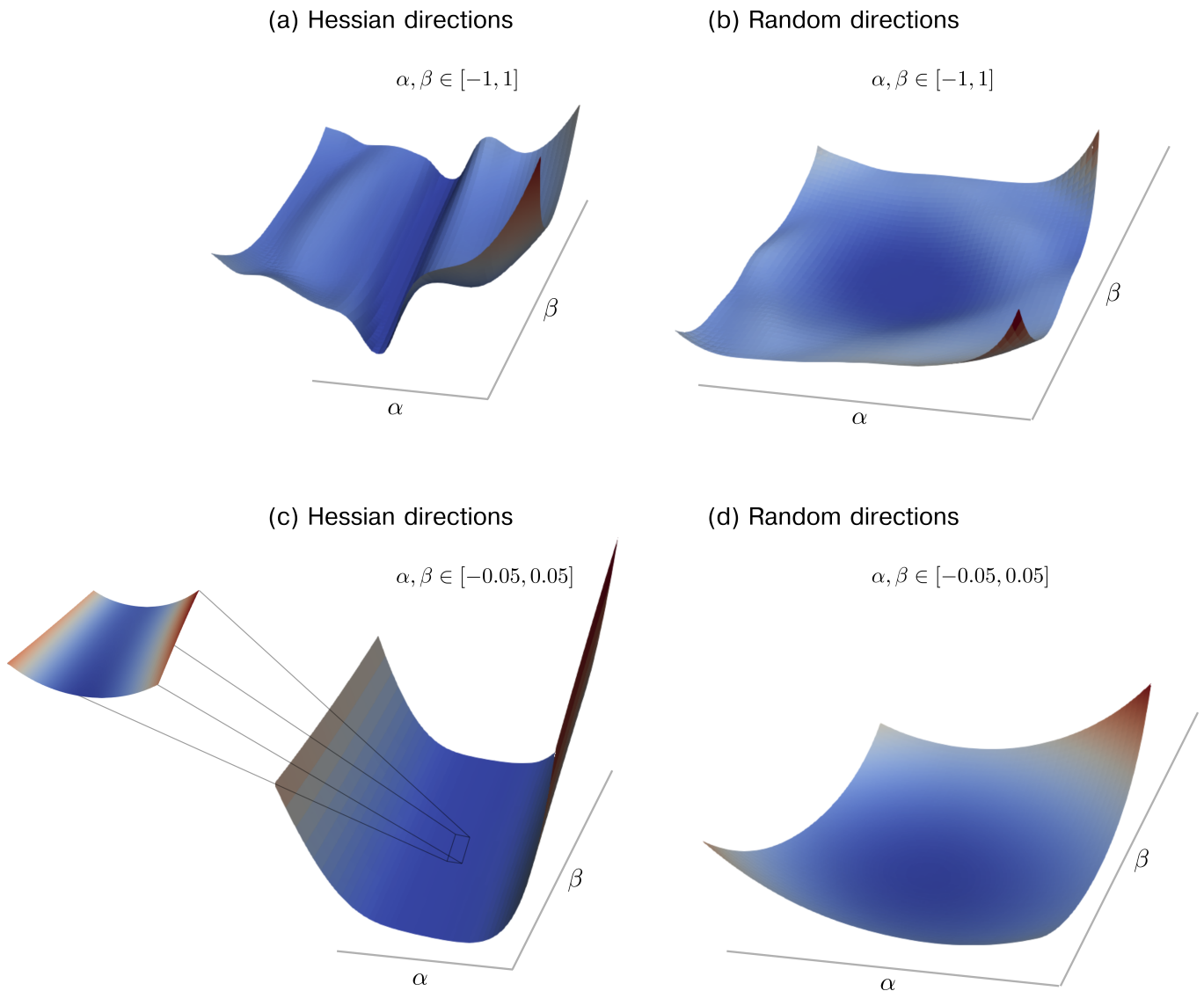


FIG. 5. Loss landscape projections for ResNet-56. (a,c) The projection directions η, δ are given by the eigenvectors associated with the largest and smallest eigenvalues of the Hessian H_θ , respectively. The zoomed inset in panel (c) shows the loss landscape for $(\alpha, \beta) \in [-0.01, 0.005] \times [-0.05, 0.05]$. We observe a decreasing loss along the negative β -axis. (b,d) The projection directions η, δ are given by random vectors. We selected a single pair of random directions η, δ (*i.e.*, no averaging over random directions has been performed). The domains of (α, β) in panels (a,b) and (c,d) are $[-0.05, 0.05] \times [-0.05, 0.05]$ and $[-1, 1] \times [-1, 1]$, respectively. All shown cross-entropy loss landscapes are based on evaluating the CIFAR-10 training dataset that consists of 50,000 images.

associated with the scalar product $[(\nabla_\theta L)^\top v]$. Because the vector v does not depend on θ (*i.e.*, $\nabla_\theta v = 0$), the result is $H_\theta v$. One may also use forward-mode AD in the second step to provide a more memory efficient implementation.⁵

We used the `LinearOperator` module that is available in the Python package `scipy` to represent HVPs. As an alternative to implementing HVPs manually, one may use the `hvp` function provided in the `torch.autograd.functional` module. Moreover, the Python package `PyHessian` [21] can be also used to evaluate Hessians of high-dimensional functions in a distributed manner. For the computation of dominant Hessian directions, we used the `scipy` function `eigsh` that is based on the implicitly restarted Lanczos method [48].

The computational cost of Algorithm 1 can be summarized as follows. For the HVPs, the computational cost is $\mathcal{O}(N)$ [49], where N is the number of neural-network parameters. To identify extremal eigenvalues along with

⁵ Different forward-mode AD functions have been made available in PyTorch since at least version 1.11, starting from March 2022.

their corresponding eigenvectors using a Lanczos-type method, one has to perform a certain number of matrix-vector multiplications. The required number of iterations depends on several factors, including the desired accuracy, the characteristics of the underlying Hessian matrix, and the choice of the initial vector.

We first focus on a ResNet-56 architecture that has been trained in [13] on CIFAR-10 using stochastic gradient descent (SGD) with Nesterov momentum. The number of parameters is 855,770. The training and test losses at the local optimum found by SGD are 9.20×10^{-4} and 0.29, respectively; the corresponding accuracies are 100.00 and 93.66, respectively.

Algorithm 1 Compute dominant Hessian directions

```

1:  $L_1 = \text{LinearOperator}((N, N), \text{matvec} = \text{hvp})$   $\triangleright$ initialize linear operator for HVP calculation
2:  $\text{eigval1}, \text{eigvec1} = \text{solve\_lm\_evp}(L_1)$   $\triangleright$ compute largest-magnitude eigenvalue and corresponding eigenvector associated with operator  $L_1$ 
3:  $\text{shifted\_hvp}(\text{vec}) = \text{hvp}(\text{vec}) - \text{eigval1} * \text{vec}$   $\triangleright$ define shifted HVP
4:  $L_2 = \text{LinearOperator}((N, N), \text{matvec} = \text{shifted\_hvp})$   $\triangleright$ initialize linear operator for shifted HVP calculation
5:  $\text{eigval2}, \text{eigvec2} = \text{solve\_lm\_evp}(L_2)$   $\triangleright$ compute largest-magnitude eigenvalue and corresponding eigenvector associated with operator  $L_2$ 
6:  $\text{eigval2} += \text{eigval1}$ 
7: if  $\text{eigval1} \geq 0$  then
8:    $\text{maxeigval}, \text{maxeigvec} = \text{eigval1}, \text{eigvec1}$ 
9:    $\text{mineigval}, \text{mineigvec} = \text{eigval2}, \text{eigvec2}$ 
10: else
11:    $\text{maxeigval}, \text{maxeigvec} = \text{eigval2}, \text{eigvec2}$ 
12:    $\text{mineigval}, \text{mineigvec} = \text{eigval1}, \text{eigvec1}$ 
13: return  $\text{maxeigval}, \text{maxeigvec}, \text{mineigval}, \text{mineigvec}$ 

```

In accordance with [13], we apply filter normalization to random directions. This and related normalization methods are often employed when generating random projections. One reason is that simply adding random vectors to parameters of a neural network loss function (or parameters of other functions) does not consider the range of parameters associated with different elements of that function. As a result, random perturbations may be too small or large to properly resolve the influence of certain parameters on a given function.

When calculating Hessian directions, we are directly taking into account the parameterization of the underlying functions that we want to visualize. Therefore, there is no need for an additional rescaling of different parts of the perturbation vector. Still, reparameterizations of a neural network can result in changes of curvature properties (see, *e.g.*, Theorem 4 in [11]).

Figure 5 shows the two-dimensional projections of the loss function (cross entropy loss) around a local critical point. The smallest and largest eigenvalues are -16.4 and 5007.9 , respectively. This means that the found critical point is a saddle with a maximum negative curvature that is more than two orders of magnitude smaller than the maximum positive curvature at that point. The saddle point is clearly visible in Figure 5(a,c). We observe in the zoomed inset in Figure 5(c) that the loss decreases along the negative β -axis.

If random directions are used, the corresponding projections indicate that the optimizer converged to a local minimum and not to a saddle point [Figure 5(b,d)]. Overall, the ResNet-56 visualizations that we show in Figure 5 exhibit structural similarities to those that we generated using the simple loss model (31) [Figure 4(d)].

As a second neural-network structure, we consider DenseNet-121 that has also been trained in [13] on CIFAR-10 using SGD with Nesterov momentum. In this neural network, the number of parameters is 6,956,298. Figure 6 shows the projections $L(\theta^* + \alpha\eta + \beta\delta)$ of the loss function (cross entropy loss) around a local optimum. The training and test losses of the local optimum found by SGD are 8.07×10^{-4} and 1.69×10^{-1} , respectively; the corresponding accuracies are 100.00 and 95.63, respectively. We again observe the typical saddle shape of L if dominant Hessian directions are used and an apparent local minimum for projections along random directions. One should keep in mind that the principal curvatures in the random direction plot are given by $\kappa_{\pm}^{\alpha,\beta}$ [see Eq. (21)]. At a critical point, the principal curvatures in dimension-reduced, random-projection space are, on average, equal to the sum of the principal curvatures in the original space. If only very few, small-magnitude negative principal curvatures are present in the original space, the expected random projections are associated with an apparent local minimum in dimension-reduced, random-projection space.

Analyzing the Hessian H_{θ} of DenseNet-121 around its local optimum, we find that the minimum and maximum principal curvatures are -109.1 and 1937.5 , respectively. In comparison to ResNet-56, the relative difference between these two curvatures is substantially smaller. In particular the largest-magnitude negative principal curvature in DenseNet-121 is by a factor of about 7 smaller than that in ResNet-56, indicating that more substantial loss improvements may be possible in the former by changing the neural-network parameters θ along the negative curvature direction.

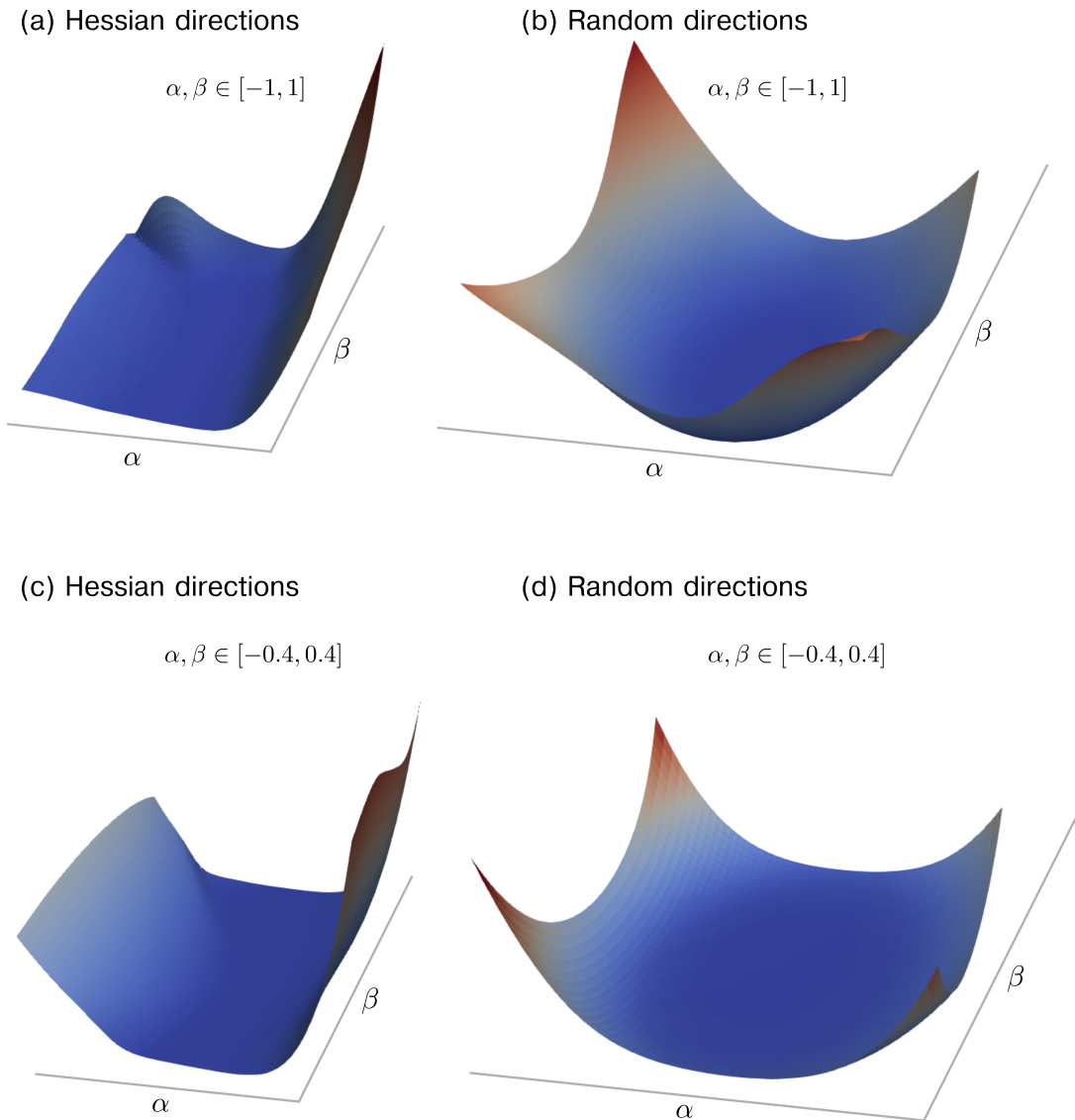


FIG. 6. Loss landscape projections for DenseNet-121. (a,c) The projection directions η, δ are given by the eigenvectors associated with the largest and smallest eigenvalues of the Hessian H_θ , respectively. (b,d) The projection directions η, δ are given by random vectors. We selected a single pair of random directions η, δ (*i.e.*, no averaging over random directions has been performed). The domains of (α, β) in panels (a,b) and (c,d) are $[-0.4, 0.4] \times [-0.4, 0.4]$ and $[-1, 1] \times [-1, 1]$, respectively. All shown cross-entropy loss landscapes are based on evaluating the CIFAR-10 training dataset that consists of 50,000 images.

In Figure 7, we compare changes in the projected loss along random and dominant Hessian directions for DenseNet-121 and ResNet-56. In addition to evaluating the loss on training data (50,000 images) as in Figures 5 and 6, we also provide a comparison to loss changes in the test dataset (10,000 images). Black crosses in Figure 7 indicate loss minima. For both neural networks, we observe that random projections are associated with loss values that increase along both directions δ, η and for both training and test data [Figure 7(a,b,e,f)]. For these projections, the loss minima are very close to the origin of the loss space. Different observations can be made if Hessian directions are employed in the loss projections. Figure 7(c) shows that the training loss minimum of DenseNet-121 in the Hessian projection is not located at the origin but at $(\alpha, \beta) \approx (0, 0.2)$. A value of $\beta \approx 0.2$ means that one has to move along the negative curvature direction to find a loss function value that is smaller than the one at the origin. We find that the value of the loss at that point is more than one order of magnitude smaller than the original loss at the origin or the smallest loss found in a random-projection plot. Similar observations can be made in the test dataset. However, the smallest loss value found in the Hessian direction plot is associated with a relatively large loss in the test dataset [Figure 7(d)]. Using Hessian directions to improve training and test performance requires one to balance potential performance improvements in the test and training datasets. Because of the relatively small

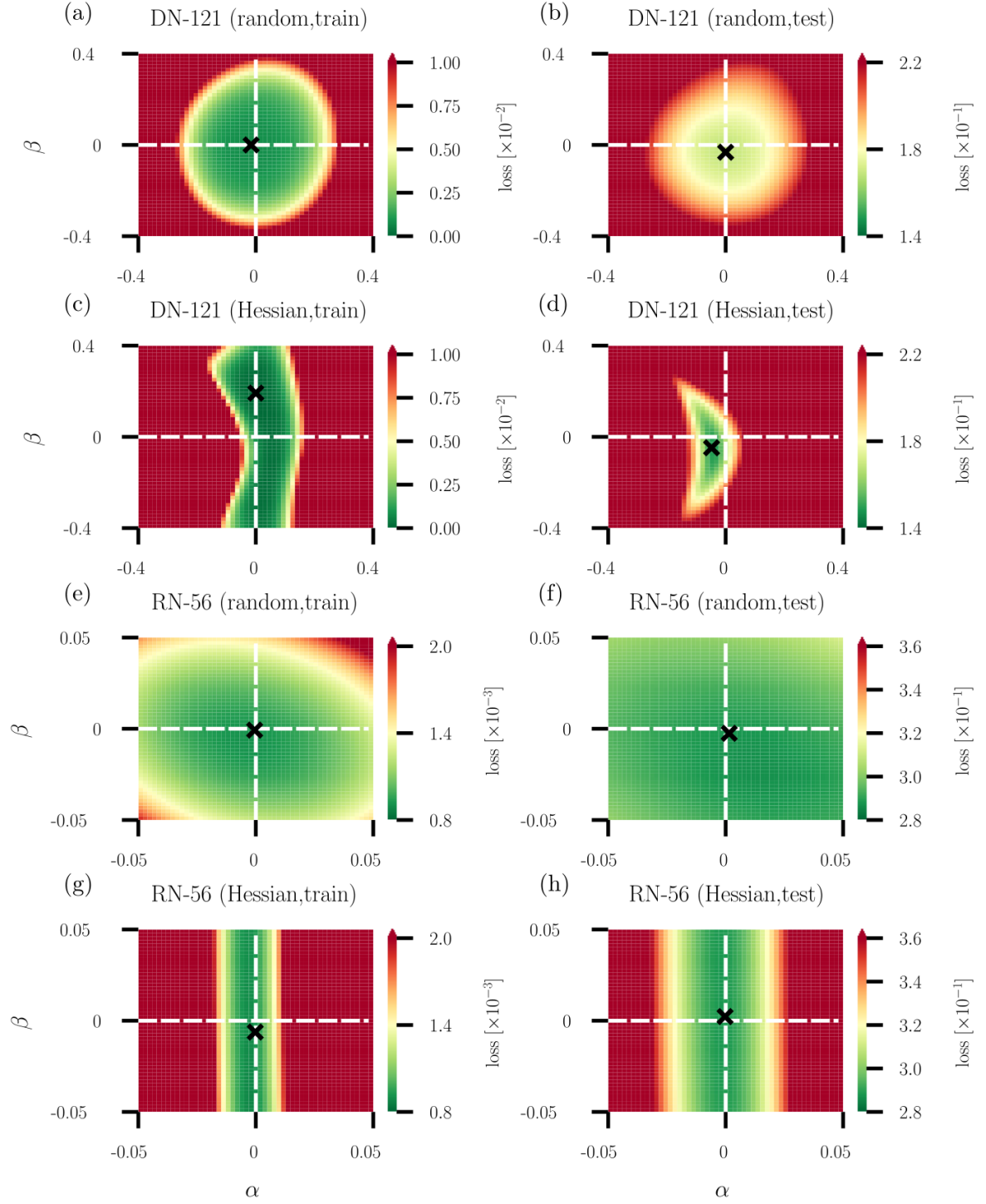


FIG. 7. Heatmaps of cross-entropy loss functions along random and dominant Hessian directions. (a–c) Loss heatmaps for DenseNet-121 [(a,c): training data, (c,d): test data]. (e–h) Loss heatmaps for ResNet-56 [(e,g): training data, (f,h): test data]. Random directions are used in panels (a,b,e,f) while Hessian directions are used in panels (c,d,g,h). Green and red regions indicate small and large loss values, respectively. Black crosses indicate the positions of loss minima in the shown domain.

negative principal curvature of ResNet-56, one can only achieve small loss improvements along the corresponding Hessian direction and differences in minimum losses between random and Hessian projections are less pronounced [Figure 7(e–h)].

In Appendix C, we study random and Hessian loss projections of three additional architectures, namely ResNet-56 without skip connections [13], MobileNetV2 [50], and GoogLeNet [50]. For ResNet-56 without skip connections, we find that the training loss minimum in the Hessian projection is associated with slightly larger training and test accuracies than the training loss minimum in the random projection. For MobileNetV2 and GoogLeNet, we find that the minimum training loss in the Hessian projection is associated with an either slightly larger training or test accuracy compared to the training loss minimum in the random projection. In the examples that we study in Appendix C, the training and test loss minima in the Hessian projections are better aligned than those in the corresponding random projections. Table II in Appendix C summarizes the results for all five architectures.

Image classification architectures and their training are already highly optimized to achieve very good performance in common image classification tasks. Our results show that it is possible to improve training loss values along the dominant negative curvature direction. Smaller loss values may be also achievable with further gradient-based training iterations, so one has to consider tradeoffs between gradient and curvature-based optimization methods. Also, assessing overall performance of dominant Hessian directions on this task is not straightforward. For instance, loss measurements and accuracy measurements are not proportional to one another for this image classification task (see Table II in Appendix C). A further complication is that, since these architectures are highly optimized and the underlying generative processes are unknown, it is difficult to interpret variations in performance that may appear between training loss, test loss, and accuracy. Therefore, to better address the question of whether better performance in training by dominant Hessian directions can translate to better test performance, we need to control for some of the suspected confounds that appear in the image classification task. So, we also study possible loss improvements along the dominant negative curvature direction in a function approximation task [20] that we describe in Appendix D.

Unlike image classification, function approximation involves sampling from a known generative distribution. Our results, summarized in Appendix D, show that both training loss and test loss are smaller in our functional approximation task for Hessian projections than for random projections. Specifically, for 2-layer and 10-layer fully connected neural networks (FCNNs), with 20,501 and 101,301 parameters, respectively, training loss is more than 9% smaller for the 2-layer network and 26% smaller for the 10-layer network for Hessian projections than for random projections and test loss is 1% smaller and 6% smaller, respectively. Using this function approximation example, we have also included in the Supplemental Information [51] an animation that shows the evolution of random and Hessian loss projections during training.

V. DISCUSSION AND CONCLUSION

Given that a global understanding of high-dimensional loss landscapes remains still very limited [1], one- and two-dimensional projections of such functions are a common tool to study the geometric properties of loss regions nearby critical points and along optimizer trajectories [12–16]. For an informed way of projecting high-dimensional functions to lower-dimensional representations, it is important to mathematically interpret the geometric information contained in such visualizations. To put it in the words of Ker-Chau Li [52]: “There are too many directions to project a high-dimensional data set and unguided plotting can be time-consuming and fruitless.” Therefore, it is important to understand how projection directions η, δ affect the resulting loss visualizations. A standard choice in the literature is to use random Gaussian directions to obtain lower-dimensional loss representations $L(\theta^* + \alpha\eta + \beta\delta)$. Combining concepts from high-dimensional probability and differential geometry, we show that saddle points in the original space may not be correctly identified as such in lower-dimensional representations if such random projections are used.

As formalized in Eq. (24), the expected principal curvature $\bar{\kappa}^{\alpha,\beta} = \text{tr}(H_\theta)$, obtained by averaging over Hessian elements $(H_\theta)_i^i$ in a dimension-reduced Hessian [see Eq. (22)], is proportional to the mean curvature $H = \text{tr}(H_\theta)/N$ in the original loss space. The connection between $\bar{\kappa}^{\alpha,\beta}$ and $\text{tr}(H_\theta)$ shows that Hutchinson-type trace estimates can be computed directly from curvature information in lower-dimensional random projections [19].

Depending on the value of the mean curvature in the original loss space, saddle points appear, on average, as either minima ($H > 0$), maxima ($H < 0$), or almost flat regions ($H \approx 0$). Instead of computing ensemble means of elements of dimension-reduced Hessians, one may also compute ensemble means of principal curvatures. As illustrated in different numerical experiments, both ways of averaging are associated with different geometric interpretations and random projections are generally not able to correctly identify saddle points. We therefore propose to study projections along dominant Hessian directions that are associated with the largest-magnitude positive and negative principal curvatures in the original space. Similar methods have been developed and applied in related work on exploratory data analysis [52]. Thus, our work may be viewed as part of a recent “Hessian turn” in machine learning characterized by computationally tractable methods for unlocking Hessian information in non-linear and high-dimensional settings

[21, 22]. Projecting high-dimensional loss functions along Hessian directions provides a genuine way of illustrating their geometric properties around critical points. Negative curvature directions may be used as a starting point for further improving the performance of a given neural network. Analyzing principal curvatures (or eigenvalues of H_θ) at critical points θ^* is also useful to identify global minima. It has been shown by Cooper [1] that for any global minimum of $L(\theta)$, the corresponding Hessian H_θ has rn positive eigenvalues, $N - rn$ eigenvalues equal to 0, and no negative eigenvalues. Here, n and r are the number of input samples (*i.e.*, data points) and the output dimension, respectively.

Our numerical experiments on different image classifiers and a function approximation task are in accordance with our reasoning and identify marked differences in the projections that are based on Hessian and random directions. Our results also address the ongoing debate on the connection between flatness of a certain loss region and generalizability (*i.e.*, the ability to effectively use trained neural networks on unseen data). Dinh *et al.* [11] provided a detailed discussion on different mechanisms (*e.g.*, reparameterization of flat regions) that indicate that sharp minima can achieve good generalization properties. Their work also highlighted the importance of clear definitions of flatness. Our work shows that flatness in randomly projected loss landscapes might be just apparent and a consequence of (large) principal curvatures of opposite sign that cancel out each other.

There are many interesting and worthwhile directions for future research building on the work reported here. To provide more insights into the geometric properties of high-dimensional loss functions of deep neural networks, future work may study other convolutional architectures [53, 54], vision transformers [55], and residual multi-layer perceptrons [56]. Our results suggest that negative curvature directions may be useful to find better local optima, but further research is needed to gain a clearer understanding of the tradeoffs associated with optimization protocols that are based on gradients and curvature information. More research on learning algorithms that minimize loss functions along negative curvature directions [57] can contribute to the development of more effective optimizers. Such guided optimization algorithms may be of particular use in applications (*e.g.*, neural network-based control [58–60] and function approximation in numerical analysis [20]) where training data can be generated with a high resolution and used to approximate a function as closely as possible. In applications where one wishes to reduce generalization error, the proposed mean-curvature estimation method may be also useful to improve existing curvature regularization approaches [61]. Furthermore, to improve our understanding of the properties of loss landscapes, it may be helpful to devote more attention to the study of connections between random matrix theory and the characterization of high-dimensional functions [62, 63]. For example, for critical points of Gaussian fields in high-dimensional spaces, Bray and Dean [62] showed that there is a monotonic relation between the index of the Hessian (*i.e.*, the number of negative eigenvalues) and the loss value: the larger the index, the larger the corresponding loss. Bray and Dean [62] also showed that the eigenvalues of the Hessian at a critical point of such fields are distributed according to Wigner’s semicircle law [64] up to an additional shift. However, traditional random-matrix-theory results such as Wigner’s semicircle law or the Marčenko–Pastur law [65] are based on various simplifying assumptions, including Gaussian weight, error, and data distributions, which do not hold in practical applications of neural networks [22].

ACKNOWLEDGEMENTS

We thank Mingtao Xia and Thomas Asikis for helpful discussions and an anonymous reviewer for their valuable feedback and suggestions.

FUNDING

LB acknowledges funding from the ARO through grant W911NF-23-1-0129.

-
- [1] Y. Cooper, Global minima of overparameterized neural networks, *SIAM Journal on Mathematics of Data Science* **3**, 676 (2021).
 - [2] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* **2**, 359 (1989).
 - [3] K. Hornik, Approximation capabilities of multilayer feedforward networks, *Neural Networks* **4**, 251 (1991).
 - [4] S. Park, C. Yun, J. Lee, and J. Shin, Minimum Width for Universal Approximation, in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021* (2021).
 - [5] M. Hardt, B. Recht, and Y. Singer, Train faster, generalize better: Stability of stochastic gradient descent, in *International Conference on Machine Learning* (Proceedings of Machine Learning Research, 2016) pp. 1225–1234.

- [6] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, The marginal value of adaptive gradient methods in machine learning, *Advances in Neural Information Processing Systems* **30** (2017).
- [7] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, On empirical comparisons of optimizers for deep learning, *International Conference on Learning Representations* (2020).
- [8] C. Baldassi, F. Pittorino, and R. Zecchina, Shaping the learning landscape in neural networks around wide flat minima, *Proceedings of the National Academy of Sciences* **117**, 161 (2020).
- [9] F. Pittorino, C. Lucibello, C. Feinauer, G. Perugini, C. Baldassi, E. Demyanenko, and R. Zecchina, Entropic gradient descent algorithms and wide flat minima, *Journal of Statistical Mechanics: Theory and Experiment* **2021**, 124015 (2021).
- [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, in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (2017).
- [11] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, Sharp Minima Can Generalize For Deep Nets, in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh (2017) pp. 1019–1028.
- [12] I. J. Goodfellow and O. Vinyals, Qualitatively Characterizing Neural Network Optimization Problems, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun (2015).
- [13] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, Visualizing the Loss Landscape of Neural Nets, in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (2018) pp. 6391–6401.
- [14] Y. Hao, L. Dong, F. Wei, and K. Xu, Visualizing and Understanding the Effectiveness of BERT, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, edited by K. Inui, J. Jiang, V. Ng, and X. Wan (Association for Computational Linguistics, 2019) pp. 4141–4150.
- [15] D. Wu, S.-T. Xia, and Y. Wang, Adversarial weight perturbation helps robust generalization, *Advances in Neural Information Processing Systems* **33**, 2958 (2020).
- [16] S. Horoi, J. Huang, B. Rieck, G. Lajoie, G. Wolf, and S. Krishnaswamy, Exploring the Geometry and Topology of Neural Network Loss Landscapes, in *Advances in Intelligent Data Analysis XX - 20th International Symposium on Intelligent Data Analysis, IDA 2022, Rennes, France, April 20-22, 2022, Proceedings*, Lecture Notes in Computer Science, Vol. 13205, edited by T. Bouadi, É. Fromont, and E. Hüllermeier (Springer, 2022) pp. 171–184.
- [17] P. Baldi and K. Hornik, Neural networks and principal component analysis: Learning from examples without local minima, *Neural Networks* **2**, 53 (1989).
- [18] Y. N. Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, and Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (2014) pp. 2933–2941.
- [19] M. F. Hutchinson, A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines, *Communications in Statistics-Simulation and Computation* **18**, 1059 (1989).
- [20] B. Adcock and N. Dexter, The gap between theory and practice in function approximation with deep neural networks, *SIAM Journal on Mathematics of Data Science* **3**, 624 (2021).
- [21] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, PyHessian: Neural networks through the lens of the Hessian, in *2020 IEEE International Conference on Big Data (Big data)* (IEEE, 2020) pp. 581–590.
- [22] Z. Liao and M. W. Mahoney, Hessian eigenspectra of more realistic nonlinear models, *Advances in Neural Information Processing Systems* **34**, 20104 (2021).
- [23] L. Böttcher, GitLab repository, <https://gitlab.com/ComputationalScience/loss-visualization> (2022).
- [24] J. M. Lee, *Riemannian Manifolds: An Introduction to Curvature*, Vol. 176 (Springer Science & Business Media, 2006).
- [25] M. Berger and B. Gostiaux, *Differential Geometry: Manifolds, Curves, and Surfaces*, Vol. 115 (Springer Science & Business Media, 2012).
- [26] W. Kühnel, *Differential Geometry*, Vol. 77 (American Mathematical Society, 2015).
- [27] K. Crane, Discrete differential geometry: An applied introduction, *Notices of the AMS, Communication* , 1153 (2018).
- [28] J. Martens, New insights and perspectives on the natural gradient method, *Journal of Machine Learning Research* **21**, 146:1 (2020).
- [29] G. Casella and R. L. Berger, *Statistical inference, 2nd edition* (Duxbury Press, Pacific Grove, CA, 2002).
- [30] E. L. Lehmann and G. Casella, *Theory of point estimation, 2nd edition* (Springer-Verlag, New York, NY, USA, 2006).
- [31] S. S. Rao, Advanced statistical inference, Texas A & M University, Lecture Notes (2017).
- [32] S.-I. Amari, Natural gradient works efficiently in learning, *Neural Computation* **10**, 251 (1998).
- [33] R. Karakida, S. Akaho, and S.-i. Amari, Universal statistics of Fisher information in deep neural networks: mean field approach, *Journal of Statistical Mechanics: Theory and Experiment* **2020**, 124005 (2020).
- [34] S.-i. Amari, H. Park, and K. Fukumizu, Adaptive method of realizing natural gradient learning for multilayer perceptrons, *Neural Computation* **12**, 1399 (2000).
- [35] H. Park, S.-I. Amari, and K. Fukumizu, Adaptive natural gradient learning algorithms for various stochastic models, *Neural Networks* **13**, 755 (2000).

- [36] R. Hecht-Nielsen, Context vectors: general purpose approximate meaning representations self-organized from raw data, *Computational Intelligence: Imitating Life*, IEEE Press, 43 (1994).
- [37] B. Laurent and P. Massart, Adaptive estimation of a quadratic functional by model selection, *Annals of Statistics*, 1302 (2000).
- [38] N. N. Schraudolph, Fast curvature matrix-vector products for second-order gradient descent, *Neural Computation* **14**, 1723 (2002).
- [39] M. Diehl and F. Messerer, Local convergence of generalized Gauss-Newton and sequential convex programming, in *58th IEEE Conference on Decision and Control, CDC 2019, Nice, France, December 11-13, 2019* (IEEE, 2019) pp. 3942–3947.
- [40] R. B. Davies, Algorithm AS 155: The distribution of a linear combination of χ^2 random variables, *Applied Statistics*, 323 (1980).
- [41] J. Bausch, On the efficient calculation of a linear combination of chi-square random variables with an application in counting string vacua, *Journal of Physics A: Mathematical and Theoretical* **46**, 505202 (2013).
- [42] T. Chen and T. Lumley, Numerical evaluation of methods approximating the distribution of a large quadratic form in normal variables, *Computational Statistics & Data Analysis* **139**, 75 (2019).
- [43] J. Matoušek, On variants of the johnson-lindenstrauss lemma, *Random Structures and Algorithms* **33**, 142 (2008).
- [44] Z. Yao, A. Gholami, K. Keutzer, and M. K. Mahoney, Hessian-based analysis of large batch training and robustness to adversaries, in *Advances in Neural Information Processing Systems (NIPS 2018)*, Vol. 31, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018) pp. 4954–4964.
- [45] Z. Bai, G. Fahey, and G. Golub, Some large-scale matrix computation problems, *Journal of Computational and Applied Mathematics* **74**, 71 (1996).
- [46] H. Avron and S. Toledo, Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix, *Journal of the ACM (JACM)* **58**, 1 (2011).
- [47] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis; 10th edition* (Cengage Learning, Boston, MA, 2015).
- [48] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods* (SIAM, 1998).
- [49] B. A. Pearlmutter, Fast exact multiplication by the Hessian, *Neural Computation* **6**, 147 (1994).
- [50] H. Phan, Pretrained TorchVision models on CIFAR-10 dataset, https://github.com/huyvnphan/PyTorch_CIFAR10 (2022).
- [51] L. Böttcher, Video of loss evolution, <https://vimeo.com/787174225> (2023).
- [52] K.-C. Li, On principal hessian directions for data visualization and dimension reduction: Another application of Stein's lemma, *Journal of the American Statistical Association* **87**, 1025 (1992).
- [53] S. Zagoruyko and N. Komodakis, Wide Residual Networks, in *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*, edited by R. C. Wilson, E. R. Hancock, and W. A. P. Smith (BMVA Press, 2016).
- [54] N. Martinel, G. L. Foresti, and C. Micheloni, Aggregating Deep Pyramidal Representations for Person Re-Identification, in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019* (Computer Vision Foundation / IEEE, 2019) pp. 1544–1554.
- [55] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021* (OpenReview.net, 2021).
- [56] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, et al., ResMLP: Feedforward networks for image classification with data-efficient training, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).
- [57] G. Alain, N. L. Roux, and P.-A. Manzagol, Negative eigenvalues of the Hessian in deep neural networks, arXiv preprint arXiv:1902.02366 (2019).
- [58] T. Asikis, L. Böttcher, and N. Antulov-Fantulin, Neural ordinary differential equation control of dynamics on graphs, *Physical Review Research* **4**, 013221 (2022).
- [59] L. Böttcher, N. Antulov-Fantulin, and T. Asikis, AI Pontryagin or how artificial neural networks learn to control dynamical systems, *Nature Communications* **13**, 333 (2022).
- [60] L. Böttcher, Gradient-free training of neural ODEs for system identification and control using ensemble Kalman inversion, in *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems* (2023).
- [61] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard, Robustness via curvature regularization, and vice versa, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 9078–9086.
- [62] A. J. Bray and D. S. Dean, Statistics of critical points of Gaussian fields on large-dimensional spaces, *Physical Review Letters* **98**, 150201 (2007).
- [63] J. W. Baron, Eigenvalue spectra and stability of directed complex networks, *Physical Review E* **106**, 064302 (2022).
- [64] E. P. Wigner, On the statistical distribution of the widths and spacings of nuclear resonance levels, in *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 47 (Cambridge University Press, 1951) pp. 790–798.
- [65] V. A. Marčenko and L. A. Pastur, Distribution of eigenvalues for some sets of random matrices, *Mathematics of the USSR-Sbornik* **1**, 457 (1967).

Appendix A: Concentration inequality

For $\eta_i, \delta_i \sim \mathcal{N}(0, 1)$ and $X_i = \eta_i + \delta_i, Y_i = \eta_i - \delta_i$, our goal is to construct a concentration inequality for $|N^{-1} \sum_i \eta_i \delta_i| = |(2N)^{-1} \sum_i X_i^2 - 1 + 1 - Y_i^2| = |(\tilde{X} - 1 + 1 - \tilde{Y})/2|$, where $\tilde{X} = N^{-1} \sum_{i=1}^N X_i^2$ and $\tilde{Y} = N^{-1} \sum_{i=1}^N Y_i^2$. Notice that \tilde{X} and \tilde{Y} are independent because the covariance associated with the corresponding bivariate normal distribution, which is valid in the limit of large N , vanishes. Using that the cumulative distribution function of the sum of independent random variables is given by the convolution of the individual distributions, we obtain

$$\begin{aligned} \Pr\left(\frac{1}{N} \sum_{i=1}^N \eta_i \delta_i \geq \epsilon\right) &= \Pr\left(\frac{1}{2N} \sum_{i=1}^N X_i^2 - 1 + 1 - Y_i^2 \geq \epsilon\right) \\ &= \Pr\left(\tilde{X} - 1 + 1 - \tilde{Y} \geq 2\epsilon\right) \\ &= \int_0^\infty \Pr\left(\tilde{X} - 1 \geq 2\epsilon + \tilde{Y} - 1 \mid \tilde{Y} = y\right) f_{\tilde{Y}}(y) dy \\ &\leq \int_0^\infty e^{-\frac{N(2\epsilon+y-1)^2}{4}} f_{\tilde{Y}}(y) dy, \end{aligned} \tag{A1}$$

where we used inequality (15) for $\epsilon \rightarrow 0$ in the last step. In the limit of large N , the distribution function $f_{\tilde{Y}}(y)$ of the random variable \tilde{Y} approaches a Gaussian with mean 1 and variance $2/N$. With this approximation, we find

$$\Pr\left(\frac{1}{N} \sum_{i=1}^N \eta_i \delta_i \geq \epsilon\right) \leq \frac{e^{-\frac{N\epsilon^2}{2}}}{\sqrt{2}}. \tag{A2}$$

The same bound applies to $\Pr\left(1 - \tilde{X} + \tilde{Y} - 1 \geq 2\epsilon\right)$, so we have

$$\Pr\left(\left|\frac{1}{N} \sum_{i=1}^N \eta_i \delta_i\right| \geq \epsilon\right) \leq \sqrt{2} e^{-\frac{N\epsilon^2}{2}}. \tag{A3}$$

For two independent random variables X, Y , a similar bound can be derived using the inequality

$$\begin{aligned} \Pr(X + Y \geq \epsilon) &= \int_{\mathbb{R}^2} \mathbf{1}_{x+y \geq \epsilon} f_{X,Y}(x, y) dx dy \\ &= \int_{\mathbb{R}^2} \mathbf{1}_{x+y \geq \epsilon} \mathbf{1}_{x \geq \epsilon/2} f_{X,Y}(x, y) dx dy \\ &\quad + \int_{\mathbb{R}^2} \mathbf{1}_{x+y \geq \epsilon} \mathbf{1}_{x < \epsilon/2} f_{X,Y}(x, y) dx dy \\ &\leq \int_{-\infty}^\infty \mathbf{1}_{x \geq \epsilon/2} f_X(x) dx + \int_{-\infty}^\infty \mathbf{1}_{y \geq \epsilon/2} f_Y(y) dy \\ &= \Pr(X \geq \epsilon/2) + \Pr(Y \geq \epsilon/2), \end{aligned} \tag{A4}$$

where $f_{X,Y}(x, y) = f_X(x)f_Y(y)$ denotes the joint distribution associated with random variables X, Y . Using inequality (A4), we obtain

$$\Pr\left(\left|\frac{1}{N} \sum_{i=1}^N \eta_i \delta_i\right| \geq \epsilon\right) \leq 4e^{-\frac{N\epsilon^2}{4}}. \tag{A5}$$

Appendix B: Annihilation method

Let $A \in \mathbb{R}^{N \times N}$ be a symmetric matrix with eigenvalues $\{\lambda_i\}_{i=1, \dots, N}$ satisfying

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_N| \quad \text{and} \quad |\lambda_k - \lambda_1| > |\lambda_j - \lambda_1| \quad \text{for } j \neq k. \tag{B1}$$

Here, λ_k is the largest-magnitude eigenvalue that belongs to the set of eigenvalues of opposite sign of λ_1 . An example of a matrix with eigenvalues that satisfy Eq. (B1) is a Hessian H_θ at a saddle point with distinct largest-magnitude

eigenvalues (*i.e.*, principal curvatures). Because the matrix A is symmetric, it has an orthonormal eigenbasis which we denote by $\{v_j\}_{j=1,\dots,N}$. Furthermore, let $B = A - \lambda_1 \mathbb{1}_{N \times N}$. For a vector $z_0 \in \mathbb{R}^N$, we calculate

$$z_1 \equiv Bz_0 = Az_0 - \lambda_1 z_0 = \sum_{j=1}^N c_j A v_j - \lambda_1 \sum_{j=1}^N c_j v_j = \sum_{j=2}^N c_j (\lambda_j - \lambda_1) v_j, \quad (\text{B2})$$

where $c_j = \langle v_j, z_0 \rangle$. The corresponding Rayleigh quotient associated with $z_n = Bz_{n-1}$ ($n \in \{1, 2, \dots\}$) is

$$\lambda^{(n)} \equiv \frac{z_n^\top B z_n}{z_n^\top z_n} = \frac{\sum_{j=2}^N c_j^2 (\lambda_j - \lambda_1)^{2n+1}}{\sum_{j=2}^N c_j^2 (\lambda_j - \lambda_1)^{2n}}, \quad (\text{B3})$$

where we used that $\langle v_i, v_j \rangle = 0$ if $i \neq j$ and 1 otherwise. Notice that $\lambda^{(n)} \rightarrow \lambda_k - \lambda_1$ in the limit $n \rightarrow \infty$. The described method is reminiscent of eigenvalue annihilation as commonly used in deflation techniques [47]. The main difference of the outlined deflation method w.r.t. standard annihilation approaches is that we are not interested in the second largest-magnitude eigenvalue, but in the largest-magnitude eigenvalue that belongs to the set of eigenvalues of opposite sign of λ_1 . The eigenvalues λ_1, λ_k can be identified with the largest-magnitude negative and positive principal curvatures and the corresponding eigenvectors are the dominant Hessian directions.

Appendix C: Image classification

To complement our results on loss landscape visualizations of common image classifiers (see Section IV), we consider three additional architectures, namely the ResNet-56 architecture from the main text but without skip connections [13], MobileNetV2 [50], and GoogLeNet [50]. The numbers of parameters of these three networks are 853,018, 2,236,682, and 5,490,986, respectively. As in the main text, all architectures were trained using SGD with Nesterov momentum. Further implementation details are available in [13, 50].

A complete summary of results for all five image classifiers is given in Table II. For ResNet-56 without skip connections, MobileNetV2, and GoogLeNet, we show heatmaps of the corresponding cross-entropy loss along random and dominant Hessian directions in Figure 8. Overall, Table II shows that random projections and Hessian directions are comparable in performance, as measured by minimum loss and maximum accuracy. However, the heatmaps in Figure 8 reveal differences between the two methods: first, in terms of the loss landscape around a learned minimizer; second, and more importantly, in terms of the degree of alignment between the location of the optimal training parameters and the location of the optimal test parameters.

The lack of skip connections in the studied ResNet-56 architecture is associated with larger loss values (training loss: 1.37×10^{-1} ; test loss: 4.33×10^{-1}) and smaller accuracies (training accuracy: 95.55; test accuracy: 86.69) compared to the ResNet-56 loss values reported in the main text. Because of the large absolute values between $10^4 - 10^8$ of the principal curvatures associated with the dominant Hessian directions, we study loss projections in a small domain $[-10^{-3}, 10^{-3}] \times [-10^{-3}, 10^{-3}]$ around the found optimum. Figure 8(a,b) shows that the minimum training and test loss points in the random projection heatmaps are not aligned. The minimum training loss is 1.36×10^{-1} with an associated accuracy of 95.58. The corresponding test loss and accuracy are 4.32×10^{-1} and 86.73, respectively. In the Hessian direction heatmaps that we show in Figure 8(c,d), the minimum training loss is 1.35×10^{-1} , slightly smaller than the minimum loss in the random-projection plot. The corresponding accuracies are slightly larger than the corresponding random projection values (training accuracy: 95.584 vs. 95.578; test accuracy: 86.74 vs. 86.73).

Also for MobileNetV2 we observe in Figure 8(e,f) that the minimum training and test loss values in the random projection heatmaps are not aligned. The minimum training loss is 4.92×10^{-2} and the corresponding test loss is 2.26×10^{-1} . The associated training and test accuracies are 99.52 and 93.90, respectively. The minimum training and test loss points in the Hessian projections in Figure 8(g,h) are more closely aligned than in the random projections. The minimum training loss in the Hessian projection is 4.90×10^{-2} , again slightly smaller than in the random projection. The training accuracy is also slightly smaller (99.516 vs. 99.524) while the corresponding test accuracy is slightly larger (93.93 vs. 93.90) compared to the random projection values.

In contrast to the other random projections in Figure 8, the GoogLeNet random projection in Figure 8(i,j) has better aligned training and test loss minima. The minimum training loss is 3.91×10^{-2} and the corresponding test loss is 2.45×10^{-1} . The associated training and test accuracies are 99.88 and 92.89. The minimum training loss in the Hessian projection is also 3.91×10^{-2} , but the training accuracy is slightly larger than in the corresponding random projection (99.884 vs. 99.878). The corresponding test accuracy is slightly smaller (92.76 vs. 92.89) than in the random projection.

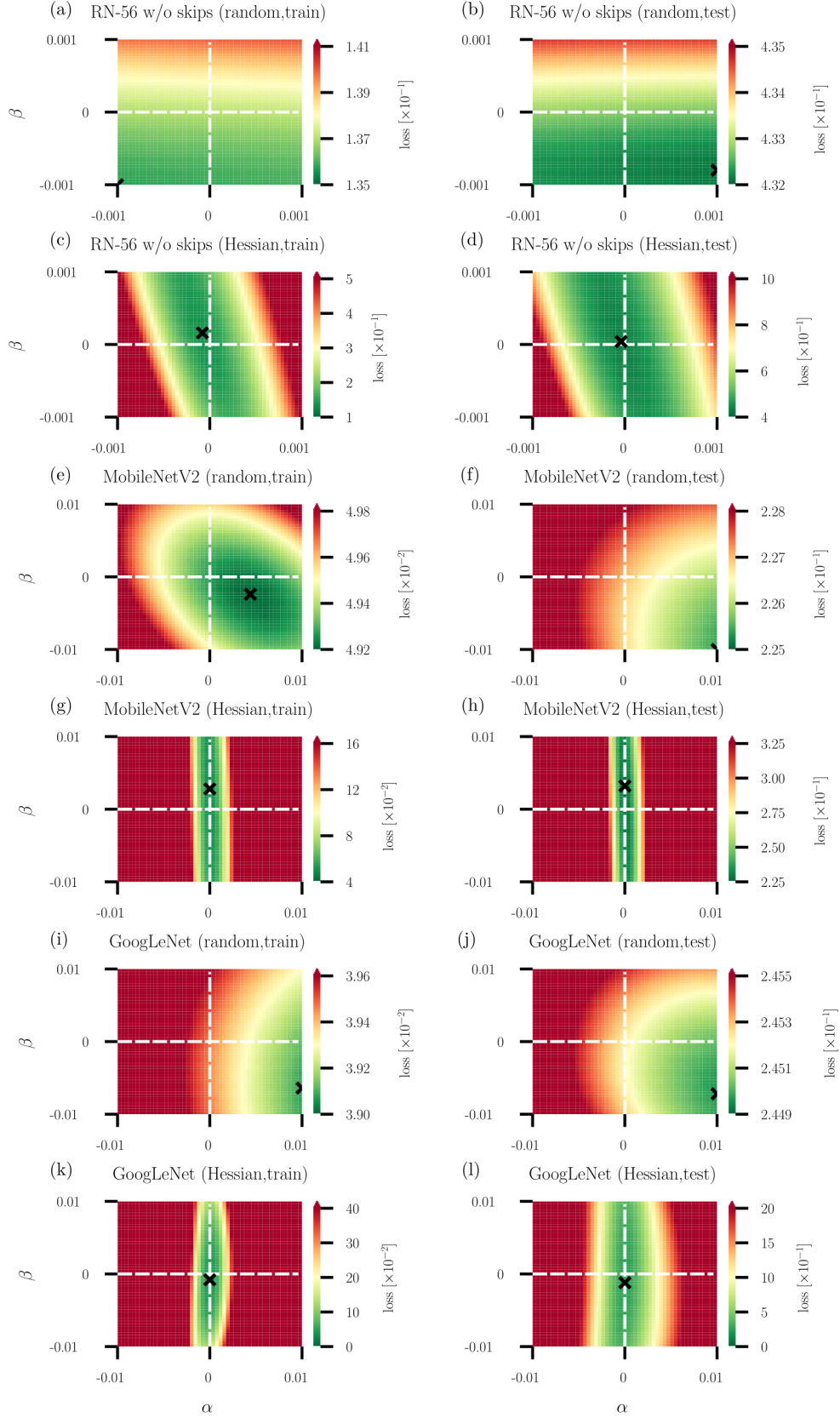


FIG. 8. Heatmaps of cross-entropy loss functions along random and dominant Hessian directions. (a–c) Loss heatmaps for ResNet-56 without skip connections [(a,c): training data, (c,d): test data]. (e–h) Loss heatmaps for MobileNetV2 [(e,g): training data, (f,h): test data]. (i–l) Loss heatmaps for GoogLeNet [(i,j): training data, (k,l): test data]. Random directions are used in panels (a,b,e,f,i,j) while Hessian directions are used in panels (c,d,g,h,k,l). Green and red regions indicate small and large loss values, respectively. Black crosses indicate the positions of loss minima in the shown domain.

	Train loss	Test loss	Train acc	Test acc	dHd
RN-56	9.20×10^{-4}	2.88×10^{-1}	99.998	93.66	$\{-16.4, 5007.9\}$
rand proj	9.10×10^{-4}	2.88×10^{-1}	99.998	93.68	–
Hess	8.75×10^{-4}	2.89×10^{-1}	99.998	93.66	–
DN-121	8.07×10^{-4}	1.69×10^{-1}	100	95.63	$\{-109.1, 1937.5\}$
rand proj	8.03×10^{-4}	1.69×10^{-1}	100	95.64	–
Hess	6.14×10^{-5}	2.46×10^{-1}	100	95.32	–
RN-56 (w/o skips)	1.37×10^{-1}	4.33×10^{-1}	95.546	86.69	$\{-39308.9, 50064044.0\}$
rand proj	1.36×10^{-1}	4.32×10^{-1}	95.578	86.73	–
Hess	1.35×10^{-1}	4.33×10^{-1}	95.584	86.74	–
MobileNetV2	4.93×10^{-2}	2.27×10^{-1}	99.518	93.91	$\{-405.0, 1061115.0\}$
rand proj	4.92×10^{-2}	2.26×10^{-1}	99.524	93.90	–
Hess	4.90×10^{-2}	2.26×10^{-1}	99.516	93.93	–
GoogLeNet	3.95×10^{-2}	2.45×10^{-1}	99.878	92.82	$\{-8208.8, 1427483.0\}$
rand proj	3.91×10^{-2}	2.45×10^{-1}	99.878	92.89	–
Hess	3.91×10^{-2}	2.44×10^{-1}	99.884	92.76	–

TABLE II. **Summary of image classification results.** We summarize the four performance measures training loss, test loss, training accuracy, and test accuracy of ResNet-56, DenseNet-121, ResNet-56 without skip connections, MobileNetV2, and GoogLeNet. For each neural network, we report all performance measures after initial training in the first row of each cell. The second and third rows in each cell list the performance measures that are associated with minimum training loss in random projections (“rand proj”) and Hessian projections (“Hess”), respectively. The column “dHd” lists the principal curvatures associated with the dominant Hessian directions at the initial training optimum.

Appendix D: Function approximation

We compare loss function visualizations that are based on random and Hessian directions in a function approximation task. In accordance with [20], we consider the smooth one-dimensional function

$$f(x) = \log(\sin(10x) + 2) + \sin(x), \quad (\text{D1})$$

where $x \in [-1, 1)$. To approximate $f(x)$, we use two fully connected neural networks (FCNNs) with 2 and 10 layers, respectively. Each layer has 100 ReLU activations. The numbers of parameters of the 2 and 10 layer architectures are 20,501 and 101,301, respectively. The training data is based on 50 points that are sampled uniformly at random from the interval $[-1, 1)$. We train both neural networks using a mean squared error (MSE) loss function and SGD with a learning rate of 0.1. The 2 and 10 layer architectures are respectively trained for 100,000 and 50,000 epochs to reach loss values of less than 10^{-4} . The best model was saved and evaluated by calculating the MSE loss for 1,000 points that were sampled uniformly at random from the interval $[-1, 1)$.

Figure 9 shows heatmaps of the training and test loss landscapes of both neural networks along random and dominant Hessian directions. Black crosses in Figure 9 indicate loss minima. As in the main text, we observe for both neural networks that random projections are associated with loss values that increase along both directions δ, η and for both training and test data [Figure 9(a,b,e,f)]. For these projections, we find that the loss minima are very close to or at the origin of the loss space. The situation is different in the loss projections that are based on Hessian directions. Figure 6(c) shows that the loss minimum for the 2-layer FCNN is not located at the origin but at $(\alpha, \beta) \approx (0, 0.04)$. We find that the value of the training loss at that point is more than 9% smaller than the smallest training loss found in a random-projection plot. The corresponding test loss is about 1% smaller than the test loss associated with the smallest training loss in the random-projection plot. For the 10-layer FCNN, the smallest training loss in the Hessian direction projection is more than 26% smaller than the smallest training loss in the randomly projected loss landscape [Figure 9(e,g)]. The corresponding test loss in the Hessian direction plot is about 6% smaller than the corresponding test loss minimum in the random direction plot [Figure 9(f,h)]. Notice that both the training and test loss minima in the random direction heatmaps in Figure 9(e,f) are located at the origin while they are located at $(\alpha, \beta) \approx (0, -0.05)$ in the Hessian direction heatmaps in Figure 9(g,h).

In the Supplemental Information [51], we include an animation that shows the evolution of lower-dimensional loss projections of the 10-layer FCNN.

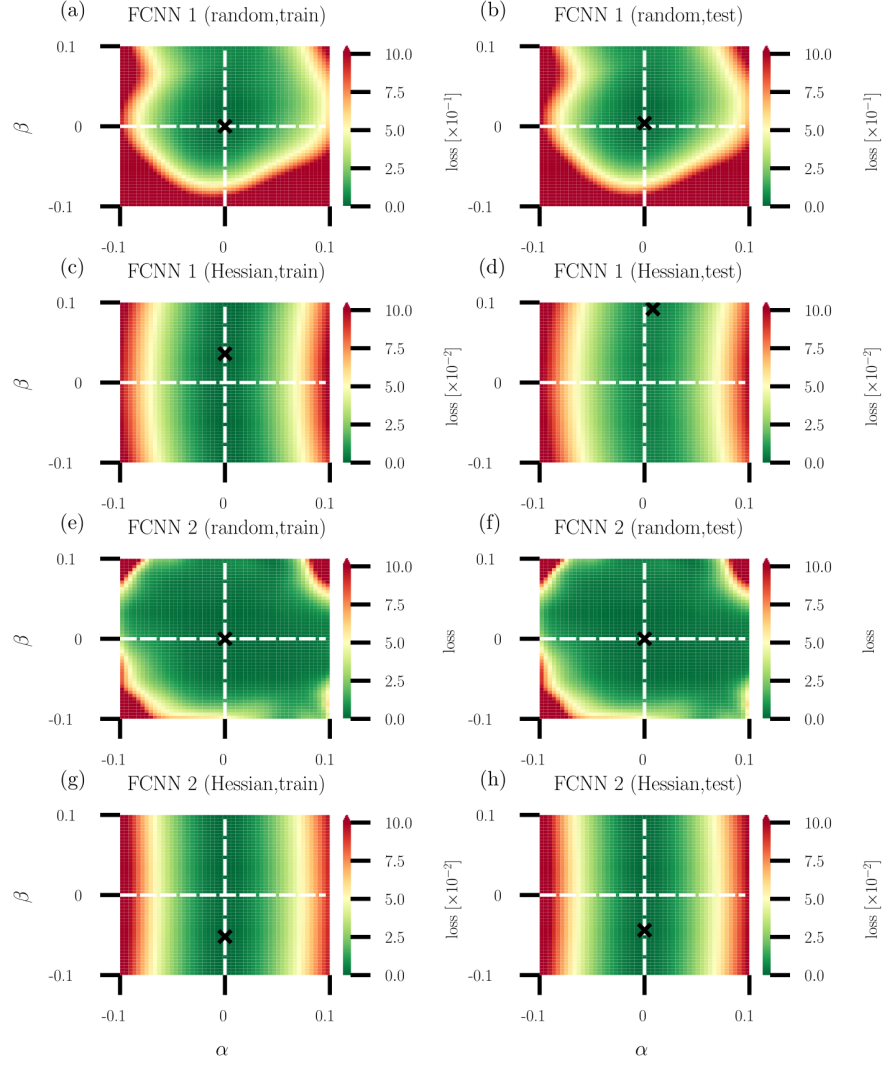


FIG. 9. Heatmaps of the mean squared error (MSE) loss along random and dominant Hessian directions for a function approximation task. (a–c) Loss heatmaps for a fully connected neural network (FCNN) with 2 layers and 100 ReLU activations per layer [(a,c): training data, (c,d): test data]. (e–h) Loss heatmaps for an FCNN with 10 layers and 100 ReLU activations per layer [(e,g): training data, (f,h): test data]. Random directions are used in panels (a,b,e,f) while Hessian directions are used in panels (c,d,g,h). Green and red regions indicate small and large mean squared error (MSE) loss values, respectively. Black crosses indicate the positions of loss minima in the shown domain. Both neural networks, FCNN 1 and FCNN 2, are trained to approximate the smooth one-dimensional function (D1).