Journal of Machine Learning Research 21 (2020) 1-64 Submitted 8/20; Revised 11/20; Published 11/20

Traces of Class/Cross-Class Structure Pervade Deep Learning Spectra

Vardan Papyan PAPYAN@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Editor: Michael Mahoney

Abstract

Numerous researchers recently applied empirical spectral analysis to the study of modern deep learning classifiers. We identify and discuss an important formal *class/cross-class structure* and show how it lies at the origin of the many visually striking features observed in deep neural network spectra, some of which were reported in recent articles, others are unveiled here for the first time. These include spectral outliers, "spikes", and small but distinct continuous distributions, "bumps", often seen beyond the edge of a "main bulk".

Keywords: deep learning, Hessian, spectral analysis, low-rank approximation, multinomial logistic regression

1. Introduction

1.1 Empirical measurements of deep neural network spectra

Recently there has been a surge of interest in measuring the spectra associated with deep classifying neural networks. LeCun et al. (2012), Dauphin et al. (2014) and Sagun et al. (2016, 2017) measured the eigenvalues of the Hessian of the parameters averaged over the training data. They plotted histograms of eigenvalues and observed a bulk, together with a few large outliers. We define these somewhat informally (see also Figure 1):

Definition 1.1 (Bulk). *A collection of eigenvalues which, when displayed in a histogram form, seemingly follows a continuous distribution.*

Definition 1.2 (Outliers). *A collection of eigenvalues, each individually isolated away from the other eigenvalues.*

Crucially, Sagun et al. (2016, 2017) observed that the number of outliers in the spectrum of the Hessian is often equal to the number of classes C. Their observation was supported by Gur-Ari et al. (2018) who noticed that the eigenvectors corresponding to these C outliers span approximately the gradients of stochastic gradient descent (SGD). Papyan (2019) developed a rigorous attribution methodology which attributed these C outliers to a rank C subspace spanned by class means of logit derivatives. Fort and Ganguli (2019) alluded to yet another related phenomenon–training deep networks is successful even when confined to

low dimensional subspace of parameters (Li et al., 2018; Jastrzębski et al., 2018b; Fort and Jastrzębski, 2019; Fort and Scherlis, 2019).

Sagun et al. (2017) experimented with: (i) two-hidden-layer networks, with 30 hidden units each, trained on synthetic data sampled from a Gaussian Mixture Model data; and (ii) one-hidden-layer networks, with 70 hidden units, trained on MNIST. Their exploration was limited to architectures with **thousands** of parameters–orders of magnitude smaller than state-of-the-art architectures such as VGG by Simonyan and Zisserman (2014) and ResNet by He et al. (2016) that have **tens of millions, hundreds of millions or close to a billion parameters** (Mahajan et al., 2018). In the absence of other deeper insights, phenomena observed in such small-scale 'academic' examples could not be expected to persist in large-scale real-world examples. In the last year, it became possible to study spectra of Hessians at full-scale. Papyan (2018) used this to observe that the patterns seen in previous small-scale examples persist even in state-of-the-art architectures.

In parallel, Ghorbani et al. (2019) studied the evolution of the full spectrum throughout the epochs of SGD, investigating the effects of skip connections, batch normalization and learning rate drops on properties of outliers and bulk. In addition to studying the spectrum of the full Hessian, Li et al. (2019) measured the spectrum of the layer-wise Fisher Information Matrix (FIM). They observed a bulk-and-outliers structure, and a closer inspection of their results shows that there is in fact more than just one bulk. Jastrzębski et al. (2020), in addition to studying the spectrum of the Hessian, also studied the spectrum of the covariance of gradients, observing a bulk-and-outliers structure in both cases. They showed how SGD hyperparameters affect the magnitude of the spectral norm and the condition number of both matrices.

There were also measurements in the literature of quantities other than the Hessian, FIM, and covariance of gradients. (Martin and Mahoney, 2018; Mahoney and Martin, 2019) measured extensively the spectrum of weight matrices throughout the layers. Their plots sometimes show a set of outliers isolated from a bulk and closer inspection suggests occasionally the presence of another small bulk beyond the main bulk. Verma et al. (2018) proposed a novel regularization scheme for training deep networks and investigated its effect on the spectra of features, which they show exhibit a bulk-and-outliers structure. Oymak et al. (2019) measured the spectrum of the backpropagated errors and showed again a set of outliers isolated from a bulk.

1.2 Initial theoretical studies

Mathematically oriented researchers tried to leverage Random Matrix Theory (RMT) to generate features similar to the ones observed in practice and study them. Pennington and Bahri (2017) decomposed the Hessian into two components, the FIM $\boldsymbol{G}$ and a residual $\boldsymbol{E}$, assumed that the eigenvalues of $\boldsymbol{G}$ are distributed according to the Marchenko-Pastur law and those of $\boldsymbol{E}$ according to the semi-circle law, and studied the predicted spectrum of the Hessian. Pennington and Worah (2018) calculated the Stieltjes transform of the spectral density of the FIM for a single hidden layer neural network with squared loss and normally distributed weights and inputs. Granziol et al. studied the deviation of the train Hessian from the population Hessian, as a function of the ratio of sample size to number of

parameters. They assumed the spectrum of the Hessian has a bulk, originating from the Gaussian Orthogonal Ensemble, with several outliers.

The loss surface of deep networks changes depending on the width of the network (Geiger et al., 2019, 2020). Mathematically oriented researchers therefore tried to leverage large-width limits to prove claims about deep neural network spectra. Karakida et al. (2019b,a) calculated the mean, variance and maximum of the FIM eigenvalues. Dyer and Gur-Ari (2019) and Andreassen and Dyer (2020) used Feynman diagrams to study the training dynamics of SGD, calculating the spectra of the Hessian and the Neural Tangent Kernel (NTK) (Jacot et al., 2018). Jacot et al. (2019a) calculated the moments of the Hessian throughout training and showed how the FIM and $\boldsymbol{E}$ are asymptotically mutually orthogonal.

At the present time, existing spectral measurements display a wide variety of features (bulk shapes, outliers, secondary mini-bulks, etc.). It seems fair to say that existing theoretical studies reproduce certain of these features. However, the connections between formal analysis and observed features are so far incomplete. In fact, it is an ongoing activity to propose generative models exhibiting the different observed phenomena.

1.3 Why are many researchers measuring deep neural network spectra?

In doing spectral analysis of each of these fundamental objects–Hessian, FIM, features, backpropagated errors, and weights–researchers hope to gain deeper insights into the behaviour of deep neural networks. Many researchers believe that such spectral features, once better understood, will provide clues to improvements in deep learning training or classifier performance (LeCun et al., 1998; Dauphin et al., 2014; Sagun et al., 2016, 2017; Gur-Ari et al., 2018; Ghorbani et al., 2019; Yao et al., 2019).

1.4 Open questions

Our goal in this work is the answer the following fundamental questions:

> **Cause attribution:** Can we say what causes outliers, mini-bulk(s) and bulk(s) in various spectra? Can we explain the number of eigenvalue outliers? Can we explain why the largest outlier is much farther out?
>
> **Ubiquity:** Why are these patterns pervasive in spectra across a variety of deep neural networks and variety of objects (features, backpropagated errors, gradients, weights, FIM, Hessian)?
>
> **Significance:** Are these patterns mere artifacts or do they convey meaningful clues? If meaningful, how can we best use the hints they give?

1.5 Insights from three-level hierarchical structure

In previous work, Papyan (2019) introduced a three-level hierarchical structure for the gradients of deep neural networks. He then introduced its connection to some of the spectral patterns in the FIM mentioned above. This work shows how this three-level hierarchical structure can be utilized to explain the spectra of all the fundamental quantities in deep learning, not just the gradients.

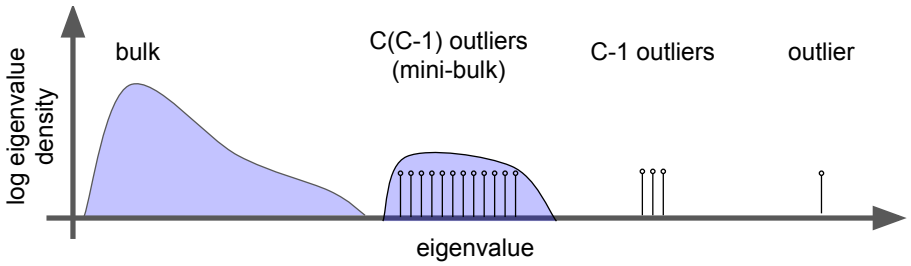


Figure 1: **Schematic typical spectrum.** In the above schematic, with C=4 classes, we see the presence of one isolated outlier on the far right, of $C-1$ secondary outliers that are less separated, of a mini-bulk consisting of $C(C-1) \approx C^2$ outliers, and of a "main bulk" on the left. These important spectral features are explained further in the body of the text.

1.6 A pattern covering all cases

We now make clear the main spectral features we will be discussing and explaining, through a pattern schematized in Figure 1. This pattern applies to any of the spectral settings mentioned earlier or any of the several new settings to be discussed below. The pattern consists of:

- A bulk;
- $C(C-1)$ eigenvalue outliers, i.e., eigenvalues outside the main bulk (for the sake of brevity we will refer to them as C^2 outliers);
- $C-1$ eigenvalue outliers situated at still larger amplitudes; and
- A single isolated outlier larger still.

The C^2 outliers may appear either as separated spikes, or alternatively as what we call a *mini-bulk*: an approximately continuous distribution rather than a series of separated spikes. We emphasize the schematic nature of the above description; the exact appearance of a spectral plot will differ from situation to situation. This schematic only applies to classification problems. Studying other tasks, such as regression, is left for future work.

1.7 Class block structure

Our first goal in this work is to explain what causes this ubiquitous pattern to emerge. To this end, we need the following definitions.

Definition 1.3 (Class block structure). *An array of vectors $\{\boldsymbol{v}_I\}_I$ exhibits a (balanced[a]) class block structure when the indices have the form $I = (i, c)$, where $1 \le i \le N$ runs across the indices of examples in a certain class, and $1 \le c \le C$ runs across the class indices.*

a. Imbalanced structure would result if different classes c had different numbers of examples per class N_c. We only study the balanced case, where $N_c = N_{c'},\ \forall c, c'$.

Example 1 (Training examples) *Training examples in standard class-balanced machine learning datasets such as CIFAR10 exhibit class block structure. For example, CIFAR10 has a total of* 50000 *training vectors, which include* 5000 *examples in class 'cat',* 5000 *examples in class 'dog', etc. We denote the i'th example in the c'th class by* $\boldsymbol{x}_{i,c}$.

Let $f(\cdot)$ be some fixed function. Consider an array $\{\boldsymbol{v}_{i,c}\}_{i,c}$, where $\boldsymbol{v}_{i,c} = f(\boldsymbol{x}_{i,c})$. Such an array inherits the class block structure from the train examples $\boldsymbol{x}_{i,c}$.

Example 2 (Features) *Consider the post-activations (also called features) at some fixed layer l of a deep neural network. They exhibit a class block structure. Indeed, they are functions of the examples and hence they inherit the class organization. The concatenation of such features across the layers also exhibits such structure. We denote the l'th layer features of* $\boldsymbol{x}_{i,c}$ *by* $\boldsymbol{h}^l_{i,c}$ *and their cross-layer concatenation by* $\boldsymbol{h}_{i,c}$.

Example 3 (Gradients) *The gradients of the loss* $\mathcal{L}(\boldsymbol{\theta})$ *with respect to the parameters of the model* $\boldsymbol{\theta}$,

$$\frac{\partial \ell(f(\boldsymbol{x}_{i,c};\boldsymbol{\theta}), \boldsymbol{y}_c)}{\partial \boldsymbol{\theta}},$$

inherit the class block structure from the examples $\boldsymbol{x}_{i,c}$.

1.8 Cross-class block structure

Assume we are using cross-entropy loss,

$$\mathcal{L}(\theta) = \operatorname*{Ave}_{i,c,c'} \mathcal{L}_{i,c,c'} = -\operatorname*{Ave}_{i,c,c'} \; y_{i,c,c'} \log\big(p_{i,c,c'}\big),$$

where $p_{i,c,c'}$ is the probability under the 'logistic' or 'softmax' model, that the i'th example in the c'th class belongs to c'. Similarly, $y_{i,c,c'}$ is the ground truth probability of the i'th example in the c'th class belonging to c', which is equal to the Kronecker delta function, $\delta_{c=c'}$. The interpretation of the second subscript (c') is different than that of the first subscript (c). The first denotes the actual class of that observation; while the second denotes the classes enumerated in applying the cross-entropy loss. In what follows, c' will generally denote such a cross-entropy class, or cross-class. c' generally represents a would-be class, as distinguished from c, the actual observed label class.

Definition 1.4 (Cross-class block structure). *An* $N \times C \times C$ *array of vectors* $\boldsymbol{v}_{i,c,c'}$ *exhibits a (balanced) cross-class block structure when it is indexed by a three-tuple,* (i, c, c'), *where* $1 \le i \le N$ *runs across the indices of examples in a certain class,* $1 \le c \le C$ *runs across the class indices, and* $1 \le c' \le C$ *runs across the cross-class indices.*

Example 4 (Losses) *The losses* $\mathcal{L}_{i,c,c'}$ *exhibit cross-class structure.*

Example 5 (Extended gradients) *The "ordinary" gradients of the loss, associated with an example* $\boldsymbol{x}_{i,c}$, *have the form:*

$$\boldsymbol{g}_{i,c,c} = \frac{\partial \ell(f(\boldsymbol{x}_{i,c};\boldsymbol{\theta}), \boldsymbol{y}_c)}{\partial \boldsymbol{\theta}} = \frac{\partial f(\boldsymbol{x}_{i,c};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}^{\top} (\boldsymbol{p}_{i,c} - \boldsymbol{y}_c);$$

they exhibit class block structure but ***not*** *cross-class block structure. We define an* ***extended gradient****, denoted* $\boldsymbol{g}_{i,c,c'}$*, which* ***does*** *exhibit cross-class block structure. In its definition, we replace in the above equation the actual observed one-hot vector* $\boldsymbol{y}_c$ *with a counterfactual one-hot vector corresponding to a would-be observation* $\boldsymbol{y}_{c'}$*:*

$$\boldsymbol{g}_{i,c,c'} = \frac{\partial \ell(f(\boldsymbol{x}_{i,c};\boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \boldsymbol{\theta}} = \frac{\partial f(\boldsymbol{x}_{i,c};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}^{\top} (\boldsymbol{p}_{i,c} - \boldsymbol{y}_{c'}).$$

Later we will see that the Fisher Information Matrix is a (weighted) second moment of extended gradients. In what follows, we refer to these extended gradients as simply gradients.

Example 6 (Backpropagated errors) *The derivative of the loss with respect to the output of some fixed layer* l*, also known as the backpropagated error, provides yet another array of vectors exhibiting cross-class block structure, provided we consider extended derivatives associated with all possible cross-class labels. In what follows, we refer to these extended backpropagated errors as simply backpropagated errors. The concatenation of backpropagated errors across layers also exhibits such structure. The* l*'th layer backpropagated error induced by example* $\boldsymbol{x}_{i,c}$ *will be denoted* $\boldsymbol{\delta}^l_{i,c,c'}$*; the cross-layer concatenation will be denoted* $\boldsymbol{\delta}_{i,c,c'}$*.*

1.9 Global mean, C class means and C^2 cross-class means

Arrays exhibiting class/cross-class block structure permit various averages to be compactly expressed.

Definition 1.5 (C^2 **cross-class means).** *For an array of vectors* $\{\boldsymbol{v}_{i,c,c'}\}_{i,c,c'}$ *exhibiting cross-class structure, we denote their* C^2 *cross-class means by* $\{\boldsymbol{v}_{c,c'}\}_{c,c'}$*; they are obtained by averaging, for a fixed class* c*, and cross-class* c'*, across the replication index* i*, i.e.,*

$$\boldsymbol{v}_{c,c'} = \operatorname*{Ave}_i \boldsymbol{v}_{i,c,c'}.$$

Example 7 *Denote the* C^2 *cross-class means of gradients by* $\{\boldsymbol{g}_{c,c'}\}_{c,c'}$*.*

Example 8 *Denote the* C^2 *cross-class means of the* l*'th layer backpropagated errors by* $\{\boldsymbol{\delta}^l_{c,c'}\}_{c,c'}$*, and their layer-wise concatenation by* $\{\boldsymbol{\delta}_{c,c'}\}_{c,c'}$*.*

Definition 1.6 (C **class means).** *For an array of vectors* $\{\boldsymbol{v}_{i,c,c'}\}_{i,c,c'}$ *exhibiting cross-class block structure, we denote their* C *class means by* $\{\boldsymbol{v}_c\}_c$*; each is obtained by averaging, for a fixed class* c*, the cross-class means associated with that class, i.e.,*

$$\boldsymbol{v}_c = \operatorname*{Ave}_{c'} \boldsymbol{v}_{c,c'} = \operatorname*{Ave}_{i,c'} \boldsymbol{v}_{i,c,c'}. \tag{1.1}$$

Moreover, an array $\{\boldsymbol{v}_{i,c}\}_{i,c}$ exhibiting class block structure has C class means, $\{\boldsymbol{v}_c\}_c$; each is obtained by averaging, for a fixed class c, across the replication index i, i.e.,

$$\boldsymbol{v}_c = \underset{i}{\text{Ave}}\, \boldsymbol{v}_{i,c}.$$

Example 9 *Denote the C feature class means at layer l by $\{\boldsymbol{h}_c^l\}_c$ and their layer-wise concatenation by $\{\boldsymbol{h}_c\}_c$.*

Example 10 *Denote the C backpropagated error class means at layer l by $\{\boldsymbol{\delta}_c^l\}_c$ and their layer-wise concatenation by $\{\boldsymbol{\delta}_c\}_c$.*

Example 11 *Denote the C class means of gradients by $\{\boldsymbol{g}_c\}_c$.*

Definition 1.7 (Global mean). *An array of vectors exhibiting class/cross-class block structure has a global mean, given by*

$$\boldsymbol{v}_G = \underset{c}{\text{Ave}}\, \boldsymbol{v}_c.$$

1.10 Second moment matrices and covariances in the class/cross-class structure

It is very natural to express second moment matrices for arrays with class/cross-class structure:

Definition 1.8 (Second moment matrix). *An array of vectors exhibiting class or cross-class block structure with D-dimensional vectors has a second moment matrix $\boldsymbol{V} \in \mathbb{R}^{D\times D}$ given by*

$$\boldsymbol{V} = \underset{i,c}{\text{Ave}}\, \boldsymbol{v}_{i,c}\boldsymbol{v}_{i,c}^\top,$$

or

$$\boldsymbol{V} = \underset{i,c,c'}{\text{Ave}}\, \boldsymbol{v}_{i,c,c'}\boldsymbol{v}_{i,c,c'}^\top,$$

respectively.

Definition 1.9 (Second moment of global mean). *Associated with an array of vectors $\{\boldsymbol{v}_{i,c,c'}\}_{i,c,c'}$ exhibiting class/cross-class structure is the second moment matrix of the global mean,*

$$\boldsymbol{v}_G\boldsymbol{v}_G^\top.$$

Definition 1.10 (Between-class second moment). *Associated with an array of vectors $\{\boldsymbol{v}_{i,c,c'}\}_{i,c,c'}$ exhibiting class/cross-class structure is the between-class second moment, $\boldsymbol{V}_{class} \in \mathbb{R}^{D\times D}$,*

$$\boldsymbol{V}_{class} = \underset{c}{\text{Ave}}\, \boldsymbol{v}_c\boldsymbol{v}_c^\top.$$

Definition 1.11 (Between-cross-class covariance). *The between-cross-class covariance,* $\boldsymbol{V}_{cross} \in \mathbb{R}^{D\times D}$ *associated with an array of vectors* $\{\boldsymbol{v}_{i,c,c'}\}_{i,c,c'}$ *exhibiting cross-class structure, is given by*

$$\boldsymbol{V}_{cross} = \underset{c,c'}{\text{Ave}}\, \boldsymbol{z}_{c,c'}\boldsymbol{z}_{c,c'}^{\top},$$

where the cross-class mean deviations $\boldsymbol{z}_{c,c'} \in \mathbb{R}^p$ *are defined as follows:*

$$\boldsymbol{z}_{c,c'} = \boldsymbol{v}_{c,c'} - \boldsymbol{v}_c.$$

Definition 1.12 (Within-cross-class covariance). *The within-cross-class covariance,*
$\boldsymbol{V}_{within} \in \mathbb{R}^{D\times D}$*, associated with an array of vectors exhibiting cross-class structure, is given by*

$$\boldsymbol{V}_{within} = \underset{i,c,c'}{\text{Ave}}\, \boldsymbol{z}_{i,c,c'}\boldsymbol{z}_{i,c,c'}^{\top},$$

where the replication deviations $\boldsymbol{z}_{i,c,c'} \in \mathbb{R}^D$ *are defined as follows:*

$$\boldsymbol{z}_{i,c,c'} = \boldsymbol{v}_{i,c,c'} - \boldsymbol{v}_{c,c'}.$$

For simplicity, the notations of mean, covariance and second moment matrix discussed so far involved averages rather than weighted averages. Below, those notations will be extended to include certain weights $w_{i,c,c'}$ associated with corresponding terms $\boldsymbol{v}_{i,c,c'}$. Moreover, we will distinguish between vectors $\boldsymbol{v}_{i,c,c'}$, where $c = c'$ and $c \neq c'$. [1]

1.11 Cause attribution

As the introduction has shown, various spectral features have been observed in the literature. By proper use of our definitions, we are able to attribute causes for all the observed features as well as new ones. The cross-entropy loss induces a three-index structure of class, cross-class and replication. This index structure–inherited by all fundamental entities in deep neural networks, including features, backpropagated errors, and (extended) gradients–allows us to easily express certain second moment and covariance matrices. We shall demonstrate empirically that these matrices cause various spectral features:

- The second moment matrix of the global mean causes the top outlier;
- The between-class covariance causes the leading cluster of $C-1$ outliers;
- The between-cross-class covariance causes the mini-bulk of $C(C-1)$ outliers; and
- The within-cross-class covariance causes the main bulk.

We will prove these assertions data-analytically by "knocking out" each of these matrices, and showing that such knockout eliminates the corresponding visual feature in the spectrum under study. We will formalize this notion of "knockout" into a formal attribution procedure.

1. In effect, we are introducing into deep networks constructs familiar in Multivariate Analysis of Variance (MANOVA), where the class/cross-class index structure would be called a two-way categorical layout. See reference (Huberty and Olejnik, 2006) for further details.

Quantity	Section	Second moment	Attribution			Figures
			C outliers	C^2 outliers (mini-bulk)	Bulk	
Hessian	5	-	$\boldsymbol{G}$	-	$\boldsymbol{E}$	3, 4
Gradients	6	$\boldsymbol{G}$	$\boldsymbol{G}_{\text{class}}$ $\boldsymbol{g}_c \approx \boldsymbol{h}_c \otimes \boldsymbol{\delta}_c$	$\boldsymbol{G}_{\text{cross}}$ $\boldsymbol{g}_{c,c'} \approx \boldsymbol{h}_c \otimes \boldsymbol{\delta}_{c,c'}$	$\boldsymbol{G}_{\text{within}}$	6
Features	7.4	$\boldsymbol{H}$	$\boldsymbol{H}_{\text{class}}$	-	$\boldsymbol{H}_{\text{within}}$	7, 8
Backprop. errors	7.7	$\boldsymbol{\Delta}$	$\boldsymbol{\Delta}_{\text{class}}$	$\boldsymbol{\Delta}_{\text{cross}}$	$\boldsymbol{\Delta}_{\text{within}}$	9, 10
Weights	7.13	$\boldsymbol{W}$	$\boldsymbol{W}_{\text{class}}$	-	$\boldsymbol{W}_{\text{within}}$	13

Table 1: **Summary of conclusions from knockout experiments.** Each row corresponds to a different quantity of interest. The column "Section" references the section in which this quantity is described and possibly decomposed into its constituent components. The column "Second moment" indicates the notation for the second moment of this quantity. The attribution columns summarize the conclusions from the knockout experiments, which attribute spectral features observed in the spectrum of this quantity. In some cases, it also provides approximations for the matrices to which the spectral features are attributed. The last "Figures" column references all figures relevant to this quantity.

1.12 Ubiquity

The effects of the class/cross-class structure permeate the spectra of features, backpropagated errors, gradients, weights, Fisher Information matrix, and Hessian, whether these are considered in the context of an individual layer or the concatenation of several layers. Specifically, we will show:

- For a fixed layer l, the Kronecker product of the c'th class mean in the features, $\boldsymbol{h}_c^l$, and the c'th class mean in the backpropagated errors, $\boldsymbol{\delta}_c^l$, approximates the c'th class mean in the gradients, $\boldsymbol{g}_c^l$.
- For a fixed layer l, the Kronecker product of the c'th class mean in the features, $\boldsymbol{h}_c^l$, and the (c, c') cross-class mean in the backpropagated errors, $\boldsymbol{\delta}_{c,c'}^l$, approximates the (c, c') cross-class mean in the gradients, $\boldsymbol{g}_{c,c'}^l$.
- Similar relations hold between the class/cross-class means of features, $\boldsymbol{h}_c$, backpropagated errors $\boldsymbol{\delta}_c$, $\boldsymbol{\delta}_{c,c'}$, and gradients $\boldsymbol{g}_c$, $\boldsymbol{g}_{c,c'}$, once these are concatenated across the layers. However, now the Kronecker product is replaced by the Khatri-Rao product of the associated quantities.
- The C class means and C^2 cross-class means in the layer-concatenated gradients induce C and C^2 outliers in the spectra of the FIM.
- Outliers in the FIM also induce outliers in the spectrum of the Hessian, as the Hessian can be written as a summation of two components, one of them being the FIM.

These insights are summarized in Figure 2, as well as Table 1.

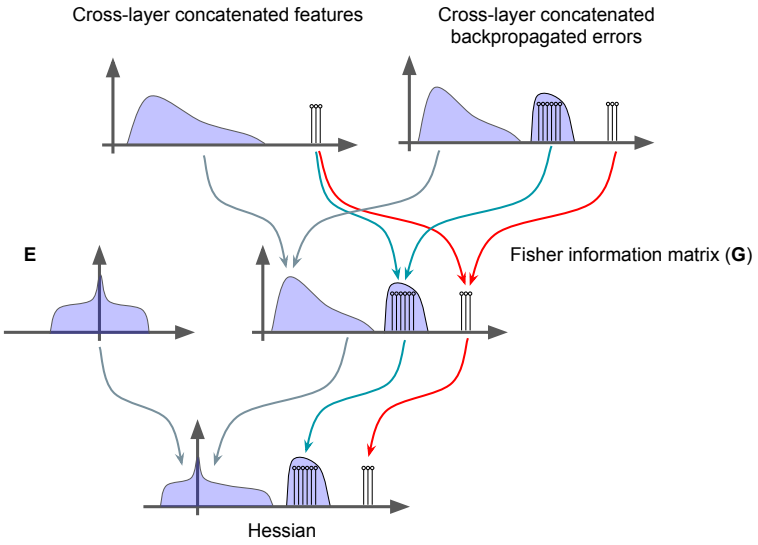


Figure 2: **Class/cross-class structure permeates all deep neural network spectra ($C = 3$ classes).** Class means in features are Khatri-Rao multiplied by class means in backpropagated errors to create class-means in FIM. Class means in features are Khatri-Rao multiplied by cross-class means in backpropagated errors to create cross-class-means in FIM. Class/cross-class means in FIM are inherited by the Hessian. This important inheritance mechanism is further explained in the body of the text.

1.13 Generalization

The outliers caused by the class means are clearly fundamental in predicting generalization. This is most evident through the following insights which will be presented in the following sections:

- In the context of multinomial logistic regression trained on Gaussian Mixture Model data, the ratio of outliers to bulk predicts misclassification.
- In the context of deep neural networks, feature class means *gradually separate* from the bulk with growing depth and also *gradually become orthogonal*. The ratio of outliers to bulk therefore predicts *layer-wise linear separability*, while the standard deviation of the outliers represents the *layer-wise orthogonality* of the classes.

1.14 Significance

There is by now a very extensive literature studying the eigenstructure of the Hessian, as well as other fundamental deep network objects (LeCun et al., 2012; Dauphin et al., 2014; Sagun et al., 2016, 2017; Pennington and Bahri, 2017; Gur-Ari et al., 2018; Pennington and Worah, 2018; Jastrzębski et al., 2018b; Martin and Mahoney, 2018; Verma et al., 2018; Jacot et al., 2018; Li et al., 2018; Karakida et al., 2019a; Dyer and Gur-Ari, 2019; Jacot et al., 2019a; Karakida et al., 2019b; Granziol et al.; Fort and Ganguli, 2019; Fort and Jastrzębski, 2019; Fort and Scherlis, 2019; Ghorbani et al., 2019; Mahoney and Martin, 2019; Oymak et al., 2019; Li et al., 2019; Jastrzębski et al., 2020; Andreassen and Dyer, 2020). These works

were authored by some of the most prominent figures in the machine learning community, who were fascinated by the structures appearing in various empirical measurements of deep network spectra. As far as we are aware, our manuscript is the first paper that explains where these structures are coming from. Part of the significance is therefore the explanation of pre-existing measurements using mathematical concepts.

At the core of mathematical statistics it is well-understood that the Hessian and FIM give fundamental inequalities for how well statistical quantities can be estimated. For the last one hundred years researchers were leveraging the Fisher information matrix – first introduced in 1924 by R. A. Fisher – for predicting generalization, one of the most well-known results being Cramer's and Rao's bound. The literature on studying the Hessian of deep networks evolved because researchers understood, in the same way Fisher understood, that the Hessian and FIM are important for predicting generalization. Dziugaite and Roy (2017) demonstrate this by showing how a PAC-Bayes bound, based on a diagonal approximation of the Hessian, can be used to predict the generalization performance of deep networks. Part of the significance of studying the spectra of the Hessian and the Fisher information matrix is to provide better estimates for these matrices, which, in turn, would translate into better generalization bounds.

2. Problem setting

Consider the balanced C-class classification problem whereby, given n training examples in each of the C different classes and their corresponding labels, the goal is to predict the labels on future data. Denote by $\boldsymbol{x}_{i,c}$ the i'th training example in the c'th class and by $\boldsymbol{y}_c$ its corresponding one-hot vector. A network is trained to classify an input $\boldsymbol{x}_{i,c}$ by passing it through a cascade of nonlinear transformations, ending with a linear classifier that outputs a set of predictions, $f(\boldsymbol{x}_{i,c};\boldsymbol{\theta}) \in \mathbb{R}^C$. The parameters of the network, denoted by $\boldsymbol{\theta} \in \mathbb{R}^p$, are trained using stochastic gradient descent (SGD) by minimizing the empirical cross-entropy loss ℓ averaged over the training data,

$$\mathcal{L}(\boldsymbol{\theta}) = \operatorname*{Ave}_{i,c} \ell(f(\boldsymbol{x}_{i,c};\boldsymbol{\theta}), \boldsymbol{y}_c).$$

The train Hessian is defined to be the second derivative of the loss with respect to the parameters of the model, averaged over the training data, i.e.,

$$\mathrm{Hess}(\theta) = \operatorname*{Ave}_{i,c} \left\{ \frac{\partial^2 \ell(f(\boldsymbol{x}_{i,c};\theta), \boldsymbol{y}_c)}{\partial \boldsymbol{\theta}^2} \right\}.$$

3. Spectral attribution via knockouts

Throughout this work we will be pointing to spectral features visible in the eigenvalue distribution of various second moment and covariance matrices. We will attribute these features to various causes. We use two particular attribution procedures, based on different notions of knockout.

Definition 3.1 (Subtraction knockout). *The process of subtracting a matrix $\boldsymbol{B}$ from another matrix $\boldsymbol{A}$ with the aim of eliminating certain spectral features. The resulting matrix will be denoted by $\boldsymbol{A} \ominus \boldsymbol{B}$.*

Definition 3.2 (Projection knockout). *The process of projecting the column space of a matrix $\boldsymbol{B}$ from another matrix $\boldsymbol{A}$, with the aim of eliminating certain spectral features. Mathematically, this is equivalent to computing $(\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^\dagger)\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^\dagger)$, where $\boldsymbol{B}^\dagger$ is the Moore–Penrose pseudoinverse of the matrix $\boldsymbol{B}$. The resulting matrix will be denoted by $\boldsymbol{A} \nparallel \boldsymbol{B}$. Assuming $\boldsymbol{A}$ and $\boldsymbol{B}$ are not square matrices, we define the projection knockout to be*

$$\boldsymbol{A} \nparallel \boldsymbol{B} = (\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{V}^\top),$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ contain all the left and right singular vectors of $\boldsymbol{B}$, respectively.

Definition 3.3 (Spectral attribution via linear algebraic knockouts). *The process of attributing spectral features in the spectrum of matrix $\boldsymbol{A}$ to the spectrum of another matrix $\boldsymbol{B}$ by observing that these spectral features visually disappear after $\boldsymbol{B}$ is knocked out.*

4. Hessian spectrum

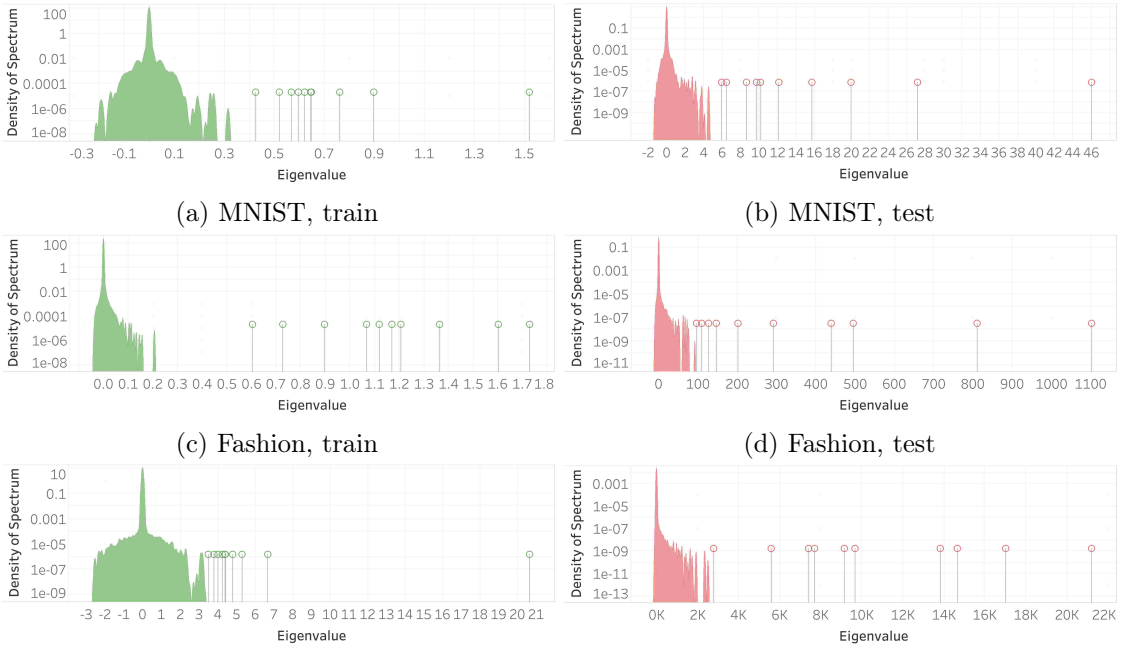
Sagun et al. (2016, 2017) measured the spectrum of the Hessian, observing a bulk-and-outliers structure with approximately C outliers. They experimented with: (i) two-hidden-layer networks, with 30 hidden units each, trained on synthetic data sampled from a Gaussian Mixture Model data; and (ii) one-hidden-layer networks, with 70 hidden units, trained on MNIST. In this section, we confirm their reports, this time at the full scale of modern state-of-the-art networks trained on real natural images.

We release software implementing state-of-the-art tools in numerical linear algebra, which allows one to approximate efficiently the spectrum of the Hessian of modern deep neural networks such as VGG and ResNet. We describe its functionality in Appendix C. Similar tools were concurrently proposed in the literature by Ghorbani et al. (2019); Pfahler and Morik (2019); Granziol et al.; Chatzimichailidis et al. (2019), each utilized for a different purpose.

4.1 Spectrum of Hessian has structure

In Figure 3 we plot the spectra of the train and test Hessian of VGG11, an architecture with 28 million parameters, trained on various datasets. The top-C eigenspace was estimated precisely using LowRankDeflation (built upon the power method) and the rest of the spectrum was approximated using LanczosApproxSpec. Both are described in Appendix C, where we also show the same plots except without first applying LowRankDeflation.

We observe a clear bulk-and-outliers structure with, arguably, C outliers. The bulk is centered around zero and there is a big concentration of eigenvalues at zero due to the large number of parameters in the model (28 million) compared to the small amount of training



(a) MNIST, train (b) MNIST, test

(c) Fashion, train (d) Fashion, test

(e) CIFAR10, train (f) CIFAR10, test

Figure 3: **Spectrum of the Hessian for VGG11 trained on various datasets.** Each row of panels documents a 'well-known' or 'standard' dataset in deep learning. The panels in the left column correspond to the train Hessian, while those in the right column to the test Hessian. Notice the presence of a bulk and $C(=10)$ outliers. The y-axis is on a logarithmic scale.

(50 thousand) or testing (10 thousand) examples. As pointed out by Sagun et al. (2016, 2017) and further discussed by Alain et al. (2019), negative eigenvalues exist in the spectrum of the train Hessian. This is despite the fact that the model was trained for hundreds of epochs, the learning rate was annealed twice and its initial value was optimized over a set of 100 values. Note there is a clear difference in magnitude between the train and test Hessian, despite the fact that both were normalized by the number of contributing terms. This phenomenon is explained by Granziol (2020).

5. Decomposing Hessian into two components:

Hess $= \boldsymbol{G} + \boldsymbol{E}$

Following the ideas presented by Sagun et al. (2016), we use the generalized Gauss-Newton decomposition of the Hessian and write it as a summation of two components:

$$\text{Hess} = \underbrace{\underset{i,c}{\text{Ave}} \left\{ \frac{\partial f(\boldsymbol{x}_{i,c};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}^\top \frac{\partial^2 \ell(\boldsymbol{z},\boldsymbol{y}_c)}{\partial \boldsymbol{z}^2}\bigg|_{\boldsymbol{z}_{i,c}} \frac{\partial f(\boldsymbol{x}_{i,c};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\}}_{\boldsymbol{G}} \tag{5.1}$$

$$+ \underbrace{\underset{i,c}{\text{Ave}} \left\{ \sum_{c'=1}^{C} \frac{\partial \ell(\boldsymbol{z},\boldsymbol{y}_c)}{\partial z_{c'}}\bigg|_{\boldsymbol{z}_{i,c}} \frac{\partial^2 f_{c'}(\boldsymbol{x}_{i,c};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2} \right\}}_{\boldsymbol{E}}, \tag{5.2}$$

where $\boldsymbol{z}_{i,c} = f(\boldsymbol{x}_{i,c};\boldsymbol{\theta})$. In mathematical statistics $\boldsymbol{G}$ is called the Fisher Information Matrix (FIM). Moreover, it is related to the natural gradient algorithm (Amari, 1998), as explained by Pascanu and Bengio (2013).

5.1 Outliers attributable to $\boldsymbol{G}$, bulk attributable to $\boldsymbol{E}$

Figure 4 plots: (i) the spectrum of the Hessian, (ii) the spectrum of the Hessian after $\boldsymbol{E}$ is knocked out via a subtraction knockout and only $\boldsymbol{G}$ is left; and (iii) the spectrum of the Hessian after $\boldsymbol{G}$ is knocked out via a subtraction knockout and only $\boldsymbol{E}$ is left. Each spectrum was approximated using LanczosApproxSpec and the LowRankDeflation procedure was applied on the Hessian and the $\boldsymbol{G}$ component to approximate the top-C subspace. Notice how the spectra of all three matrices resemble variations of Figure 1.

Notice how knocking out $\boldsymbol{E}$ shrinks the bulk significantly, indicating that the bulk originates largely from $\boldsymbol{E}$. Note also that the upper tails of the test Hessian and test $\boldsymbol{G}$ obey eigenvalue interlacing, as in Cauchy's interlacing theorem (Horn and Johnson, 2012).

Notice how knocking out $\boldsymbol{G}$ eliminates the outliers in the spectrum; the outliers in the Hessian are attributable to $\boldsymbol{G}$. Papyan (2019) showed that the outliers in $\boldsymbol{G}$ are attributable to the presence of C high magnitude gradient class means, which explains our previous observation of C outliers in the spectrum of the Hessian.

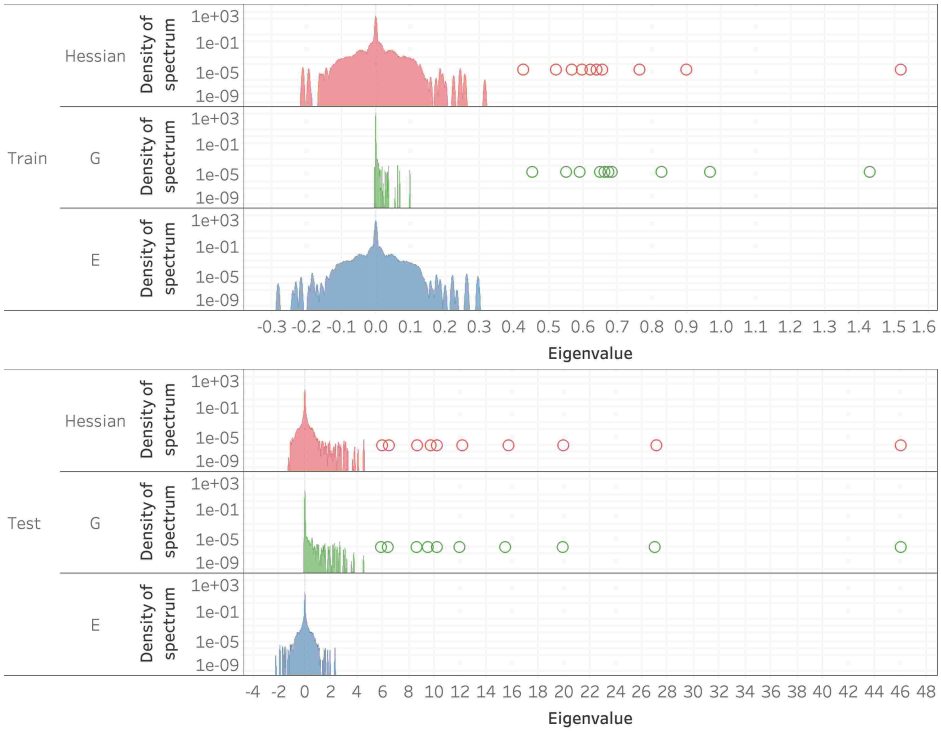
6. Cross-class structure in $\boldsymbol{G}$

Papyan (2019) proposed to decompose $\boldsymbol{G}$ based on the cross-class structure. We follow their proposition but slightly modify their decomposition. Recall the definition of an extended gradient[2]

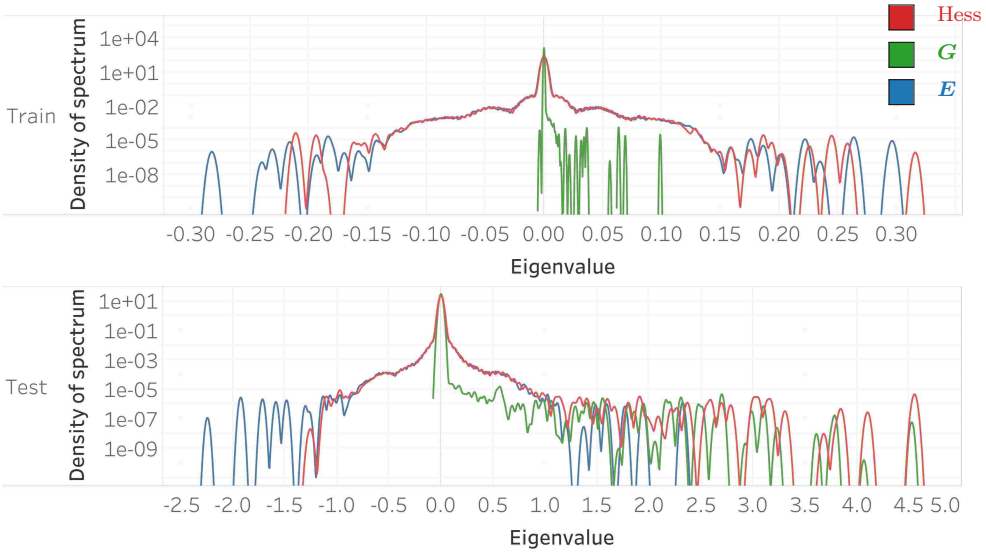
$$\boldsymbol{g}_{i,c,c'} = \frac{\partial \ell(f(\boldsymbol{x}_{i,c};\boldsymbol{\theta}),\boldsymbol{y}_{c'})}{\partial \boldsymbol{\theta}}. \tag{6.1}$$

Note that for $c = c'$, $\boldsymbol{g}_{i,c,c}$ is simply the usual gradient of the i'th training example in the c'th class. Alternatively, for $c \neq c'$, $\boldsymbol{g}_{i,c,c'}$ is the would-be gradient of the i'th training example in the c'th class, as if it belonged to cross-class c' instead. This definition is useful since, as we

2. In Papyan (2019) $\boldsymbol{g}_{i,c,c'}$ is defined slightly differently.



(a) **Outliers attributed to G.** The first block of three panels plots train spectra, while the second block plots test spectra. Within each block, each panel depicts the spectrum of a different matrix. The top panel: spectrum of Hessian. Middle panel: spectrum of $\boldsymbol{G}$ ($\boldsymbol{E}$ subtraction-knocked out). Bottom panel: spectrum of $\boldsymbol{E}$ ($\boldsymbol{G}$ subtraction-knocked out). Note how $\boldsymbol{G}$ is clearly responsible for the outliers.



(b) **Bulk attributed to E.** Zoom in on the bulks of the Hessian and its two components. The top panel plots train spectra; the bottom, test spectra.

Figure 4: **Spectrum of the Hessian with its constituent components.** The network is VGG11 and it was trained on MNIST sub-sampled to 5000 examples per class. The y-axis of all plots is on a logarithmic scale.

prove in Appendix A, $\boldsymbol{G}$ is a weighted second moment matrix of these gradients, i.e.,

$$\boldsymbol{G} = \sum_{i,c,c'} w_{i,c,c'} \boldsymbol{g}_{i,c,c'} \boldsymbol{g}_{i,c,c'}^\top, \tag{6.2}$$

where the weights $w_{i,c,c'}$ are defined as follows:

$$w_{i,c,c'} = \frac{p_{i,c,c'}}{nC}.$$

Above, $p_{i,c,c'}$ is the c'-th entry of $p(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}) \in \mathbb{R}^C$, which are the Softmax probabilities of $\boldsymbol{x}_{i,c}$. Define the *gradient cross-class means*:

$$\boldsymbol{g}_{c,c'} = \sum_i \pi_{i,c,c'} \boldsymbol{g}_{i,c,c'},$$

the *gradient within-cross-class covariance*:

$$\boldsymbol{G}_{\text{within}} = \sum_{i,c,c'} w_{i,c,c'} (\boldsymbol{g}_{i,c,c'} - \boldsymbol{g}_{c,c'})(\boldsymbol{g}_{i,c,c'} - \boldsymbol{g}_{c,c'})^\top, \tag{6.3}$$

and its class/cross-class specific versions:

$$\boldsymbol{G}_{\text{within},c,c'} = \sum_i \pi_{i,c,c'} (\boldsymbol{g}_{i,c,c'} - \boldsymbol{g}_{c,c'})(\boldsymbol{g}_{i,c,c'} - \boldsymbol{g}_{c,c'})^\top,$$

where the weights are given by

$$\pi_{i,c,c'} = \frac{w_{i,c,c'}}{w_{c,c'}}$$
$$w_{c,c'} = \sum_i w_{i,c,c'}.$$

These equations group together gradients for a fixed pair of c, c'. Define the *gradient class means*:

$$\boldsymbol{g}_c = \sum_{c' \neq c} \pi_{c,c'} \boldsymbol{g}_{c,c'}.$$

Notice that the above average does not take into account $\boldsymbol{g}_{c,c}$. The reason is that these are gradient means, which are approximately equal to zero at convergence of SGD. Define also the *between-class gradient second moment*:

$$\boldsymbol{G}_{\text{class}} = \sum_c \boldsymbol{w}_c \boldsymbol{g}_c \boldsymbol{g}_c^\top,$$

the *within-cross-class gradient covariance*:

$$\boldsymbol{G}_{\text{cross}} = \sum_{c,c' \neq c} w_{c,c'} (\boldsymbol{g}_{c,c'} - \boldsymbol{g}_c)(\boldsymbol{g}_{c,c'} - \boldsymbol{g}_c)^\top,$$

and its class-specific version:

$$\boldsymbol{G}_{\text{cross},c} = \sum_{c' \neq c} \pi_{c,c'} (\boldsymbol{g}_{c,c'} - \boldsymbol{g}_{c'})(\boldsymbol{g}_{c,c'} - \boldsymbol{g}_{c'})^\top,$$

where the weights are given by

$$\pi_{c,c'} = \frac{w_{c,c'}}{w_c}$$
$$w_c = \sum_{c' \neq c} w_{c,c'}.$$

These equations represent an even coarser grouping, where the gradients with a fixed c (and possibly varying c') are grouped together. Although not used above, we will also define $\pi_c = \frac{w_c}{\sum_c w_c}$. Leveraging these definitions, we prove in Appendix A that $\boldsymbol{G}$ can be decomposed as follows:

$$\boldsymbol{G} = \boldsymbol{G}_{\text{class}} + \boldsymbol{G}_{\text{cross}} + \boldsymbol{G}_{\text{within}} + \underbrace{\sum_{c} w_{c,c} \boldsymbol{g}_{c,c} \boldsymbol{g}_{c,c}^\top}_{\boldsymbol{G}_{c=c'}}. \tag{6.4}$$

6.1 C outliers attributable to $\boldsymbol{G}_{\text{class}}$, C^2 outliers attributable to $\boldsymbol{G}_{\text{cross}}$ and bulk attributable to $\boldsymbol{G}_{\text{within}}$

Papyan (2019) showed empirically that the outliers in the spectrum of $\boldsymbol{G}$ are attributable to the covariance of the gradient class means, i.e., $\boldsymbol{G}_{\text{class}}$ in our decomposition. However, that earlier work did not provide empirical evidence for the existence of other components in the decomposition in Equation (6.4), since these are quite subtle and not always visibly pronounced in the spectra of $\boldsymbol{G}$ or its knockouts. For the present work, we developed a tool[3] to approximate the spectrum of $\log(\boldsymbol{G})$ by leveraging the numerical linear algebra machinery developed in Appendix C. It turns out, the spectrum of $\log(\boldsymbol{G})$ rather than $\boldsymbol{G}$ exposes all the components in the decomposition.

The first panel of Figure 5 depicts the spectrum of $\log(\boldsymbol{G})$. Notice its surprisingly simple structure with three separated bulks. The one in the middle is the main bulk that would ordinarily be seen in the spectrum of $\boldsymbol{G}$. The left and right bulks have very low density of eigenvalues compared to it–they can only be seen because we are looking at the spectrum of $\log(\boldsymbol{G})$. The same Figure also shows the spectrum of $\boldsymbol{G}$ once different matrices are knocked out. Once $\boldsymbol{G}_{\text{class}}$ is knocked out, the C outliers on the right disappear, corroborating previous findings by Papyan (2019) that claim these outliers are attributable to $\boldsymbol{G}_{\text{class}}$. More importantly, once $\boldsymbol{G}_{\text{cross}}$ is knocked out, the left mini-bulk disappears; it is attributable to the between-cross-class covariance. Once $\boldsymbol{G}_{\text{within}}$ is knocked out, the main bulk disappears, implying it is attributable to the within-cross-class covariance. Subtracting $\boldsymbol{G}_{c=c'}$ has no clear effect on the spectrum, which can be explained by noticing that $\boldsymbol{G}_{c=c'}$ is a second moment of gradient means, which are approximately equal to zero at convergence of SGD.

3. The matrix $\boldsymbol{G}$ is very large – having several millions of rows and columns – and therefore its eigenvalues can not be calculated directly by simply invoking standard linear algebra software. Instead, our experiments approximate the distribution of the eigenvalues by using the Lanczos algorithm, as explained in Appendix C. Similarly, the log-eigenvalues of $\boldsymbol{G}$ can not be calculated directly by simply invoking standard linear algebra software and applying the logarithm function. Instead, in Section C.7 of the Appendix, we propose a novel modification to Lanczos that allows us to approximate the distribution of the log-eigenvalues of $\boldsymbol{G}$.

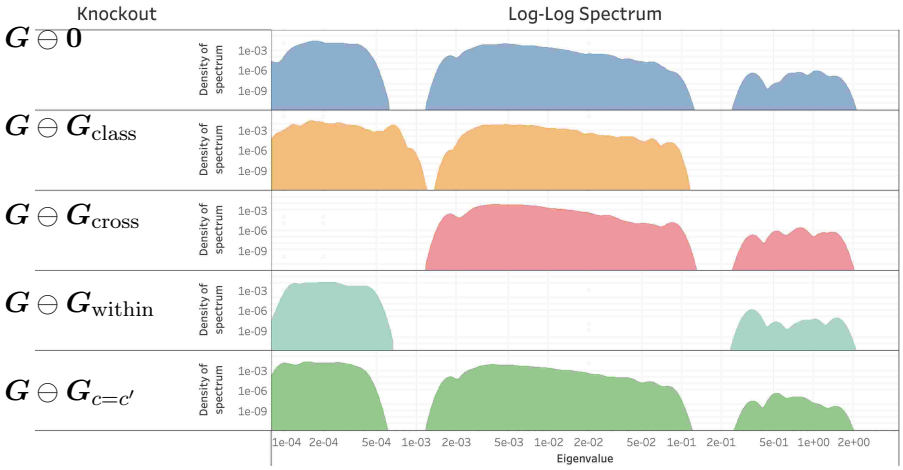


Figure 5: **Attribution via knockouts of spectral features in spectrum of G calculated on train data.** Each panel in the right column plots the approximate log spectrum of the matrix indicated in the left column. The approximation is computed using the LANCZOSAPPROXSPEC procedure. The network is VGG11 and it was trained on CIFAR10 subsampled to 136 examples per class. The y-axis is on a logarithmic scale.

6.2 Dynamics of bulk and two groups of outliers with training and sample size

Figure 6 plots spectra of $\log(\boldsymbol{G})$ across epochs and training sample size. Notice the alignment between the C outliers of $\boldsymbol{G}$, colored in orange, and the eigenvalues of $\boldsymbol{G}_{\text{class}}$, colored in yellow. Notice also the alignment between the C^2 outliers of $\boldsymbol{G}$ and the green dots corresponding to $\boldsymbol{G}_{\text{cross}}$, and also the main bulk of $\boldsymbol{G}$ and the red dots corresponding to $\boldsymbol{G}_{\text{within}}$. This correspondence aligns with the attribution of the different spectral features to the different components in the cross-class structure, observed already in the previous subsection.

Fixing sample size and increasing the number of epochs causes spectral bulks of $\boldsymbol{G}_{\text{cross}}$ and $\boldsymbol{G}_{\text{within}}$ to separate. In contrast, fixing the epoch and increasing sample size causes the spectral bulks to merge. Moreover, varying the epoch number or the training sample size does not change significantly the distance between the $\boldsymbol{G}_{\text{class}}$ and $\boldsymbol{G}_{\text{within}}$.

7. Multilayer perceptron

In this section we study the relation between the patterns in the features, backpropagated errors, gradients, FIM, and weights in the context of a multilayer perceptron (MLP), i.e., a cascade of fully connected layers.

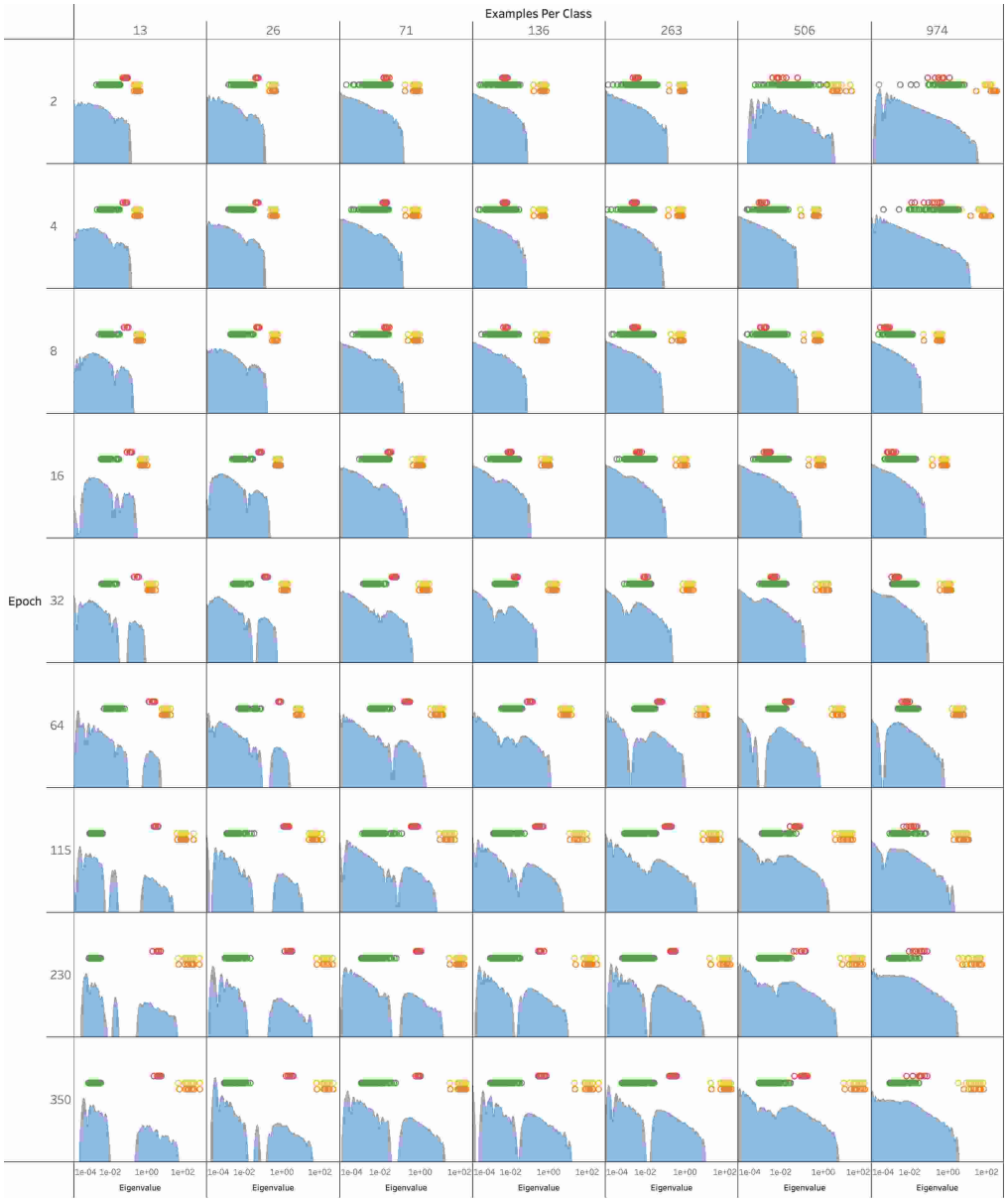


Figure 6: **Dynamics with training and sample size of cross-class structure in $\boldsymbol{G}$.** Each column of panels plots, for one specific sample size N, the spectrum of $\boldsymbol{G}$ calculated on train data throughout the epochs of SGD, so that each row corresponds to a different epoch. Each panel plots, in orange, the top-C eigenvalues of $\boldsymbol{G}$, estimated using SubspaceIteration and, in blue, the log spectrum of the rank-C deflated $\boldsymbol{G}$, estimated using LanczosApproxSpec. Each panel also plots the eigenvalues of $\boldsymbol{G}_{\text{class}} + \boldsymbol{G}_{\text{cross}} + \boldsymbol{G}_{c=c'}$. The top-$C$ eigenvalues of this matrix, which are attributable to $\boldsymbol{G}_{\text{class}}$, are colored in yellow. The next $C^2 - 2C$ eigenvalues, attributable to $\boldsymbol{G}_{\text{cross}}$, are colored in green. The final C eigenvalues, attributable to $\boldsymbol{G}_{c=c'}$, are colored in teal but are missing from the plots because their magnitude is less than 10^{-4}. Each panel also plots in red the average eigenvalue of the matrices $\{\boldsymbol{G}_{\text{within},c}\}_{c=1}^{C}$, given by $\{\frac{1}{N}\operatorname{Tr}\{\boldsymbol{G}_{\text{within},c}\}\}_{c=1}^{C}$, where $\boldsymbol{G}_{\text{within},c}$ is the within-cross-class covariance restricted to class c. The y-axis of all panels is on a logarithmic scale. For numerical stability, we approximate the spectrum of $\log\left(\boldsymbol{G} + 10^{-5}\boldsymbol{I}\right)$.

7.1 Forward pass

In the forward pass, for each layer $1 \leq l \leq L$, we multiply the features of the previous layer $\boldsymbol{h}_{i,c}^{l-1}$ by a weight[4] matrix $\boldsymbol{W}^l$ to produce the pre-activations of the next layer $\boldsymbol{z}_{i,c}^l$, i.e.,

$$\boldsymbol{z}_{i,c}^l = \boldsymbol{W}^l \boldsymbol{h}_{i,c}^{l-1}.$$

Note that for $l = 1$, $\boldsymbol{h}_{i,c}^{l-1} = \boldsymbol{h}_{i,c}^0$ is equal to $\boldsymbol{x}_{i,c}$. These are then passed through a non-linearity σ, which in our case is the rectified linear unit (ReLU), to produce the features of the next layer,

$$\boldsymbol{h}_{i,c}^l = \sigma(\boldsymbol{z}_{i,c}^l).$$

7.2 Backward pass

The backward pass computes the gradient of the loss with respect to the parameters of the model, which in this case are the weight matrices,

$$\frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_c)}{\partial \boldsymbol{W}^l}.$$

We will consider a more general gradient where the label class c and the cross-class c' are allowed to differ, i.e.,

$$\frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \boldsymbol{W}^l}.$$

Using the chain rule of calculus, one can show that:

$$\frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \boldsymbol{W}^l} = \frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \boldsymbol{z}_{i,c}^l} \frac{\partial \boldsymbol{z}_{i,c}^l}{\partial \boldsymbol{W}^l} \tag{7.1}$$

$$= \boldsymbol{\delta}_{i,c,c'}^l {\boldsymbol{h}_{i,c}^{l-1}}^\top, \tag{7.2}$$

where $\boldsymbol{\delta}_{i,c,c'}^l$ denote the *backpropagated errors*, which we define as follows:

$$\boldsymbol{\delta}_{i,c,c'}^l = \frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \boldsymbol{z}_{i,c}^l}.$$

Let $\mathrm{vec}(\cdot)$ denote the operator forming a vector from a matrix by stacking its columns as subvector blocks within a single vector. Using this operator, the above equation can be vectorized as follows:

$$\frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \,\mathrm{vec}(\boldsymbol{W}^l)} = \boldsymbol{h}_{i,c}^{l-1} \otimes \boldsymbol{\delta}_{i,c,c'}^l, \tag{7.3}$$

where $\otimes$ denotes the Kronecker product. Recall our definition of $\boldsymbol{g}_{i,c,c'}$ in Equation (6.1), and define its analogous layer-specific quantity:

$$\boldsymbol{g}_{i,c,c'}^l = \frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \,\mathrm{vec}(\boldsymbol{W}^l)};$$

4. In practice, we use batch normalization layers prior to ReLU. However, for simplicity of exposition we ignore them.

using Equation (7.3), this is equal to

$$\boldsymbol{g}^l_{i,c,c'} = \boldsymbol{h}^{l-1}_{i,c} \otimes \boldsymbol{\delta}^l_{i,c,c'}. \tag{7.4}$$

Note that $\boldsymbol{g}^l_{i,c,c'}$ is the l'th subvector of $\boldsymbol{g}_{i,c,c'}$. Define the *feature* $\boldsymbol{h}_{i,c}$ to be the concatenation of all the feature subvectors $\{\boldsymbol{h}^{l-1}_{i,c}\}_{l=1}^L$ into a single vector so that $\boldsymbol{h}^{l-1}_{i,c}$ is the l'th subvector of $\boldsymbol{h}_{i,c}$. Similarly, define the *backpropagated error* $\boldsymbol{\delta}_{i,c,c'}$ to be the concatenation of all the backpropagated errors $\{\boldsymbol{\delta}^l_{i,c,c'}\}_{l=1}^L$ into a single vector so that $\boldsymbol{\delta}^l_{i,c,c'}$ is the l'th subvector of $\boldsymbol{\delta}_{i,c,c'}$. Using these definitions, we can write

$$\boldsymbol{g}_{i,c,c'} = \boldsymbol{h}_{i,c} \odot \boldsymbol{\delta}_{i,c,c'}, \tag{7.5}$$

where $\odot$ denotes the Khatri-Rao[5] product, which computes in the l'th layer block of $\boldsymbol{g}_{i,c,c'}$ the Kronecker product of the corresponding l'th layer blocks from $\boldsymbol{\delta}_{i,c,c'}$ and $\boldsymbol{h}_{i,c}$.

7.3 Kronecker structure in $\boldsymbol{G}$

Plugging Equation (7.5) into Equation (6.2), we obtain that

$$\boldsymbol{G} = \sum_{i,c,c'} w_{i,c,c'} \boldsymbol{g}_{i,c,c'} \boldsymbol{g}^\top_{i,c,c'} \tag{7.6}$$

$$= \sum_{i,c,c'} w_{i,c,c'} (\boldsymbol{h}_{i,c} \odot \boldsymbol{\delta}_{i,c,c'})(\boldsymbol{h}_{i,c} \odot \boldsymbol{\delta}_{i,c,c'})^\top \tag{7.7}$$

$$= \sum_{i,c,c'} w_{i,c,c'} (\boldsymbol{h}_{i,c} \boldsymbol{h}^\top_{i,c}) \odot (\boldsymbol{\delta}_{i,c,c'} \boldsymbol{\delta}^\top_{i,c,c'}). \tag{7.8}$$

Similarly, the subset of $\boldsymbol{G}$ corresponding to the l'th layer is given by

$$\boldsymbol{G}^l = \sum_{i,c,c'} w_{i,c,c'} (\boldsymbol{h}^{l-1}_{i,c} \boldsymbol{h}^{l-1\top}_{i,c}) \otimes (\boldsymbol{\delta}^l_{i,c,c'} \boldsymbol{\delta}^{l\top}_{i,c,c'}).$$

The above equation relates $\boldsymbol{G}$ to the backpropagated errors $\boldsymbol{\delta}_{i,c,c'}$ and the features $\boldsymbol{h}_{i,c}$. As such, in the next subsection we study the second moment of the features, and in the following subsection the weighted second moment of the backpropagated errors.

7.4 Class block structure in features

Consider the second moment of the l'th layer features,

$$\boldsymbol{H}^l = \operatorname*{Ave}_{i,c} \boldsymbol{h}^l_{i,c} \boldsymbol{h}^{l\top}_{i,c}.$$

Define the *feature class means*:

$$\boldsymbol{h}^l_c = \operatorname*{Ave}_i \boldsymbol{h}^l_{i,c},$$

and the *feature global mean*:

$$\boldsymbol{h}^l_G = \operatorname*{Ave}_c \boldsymbol{h}^l_c.$$

5. The Khatri-Rao product is a section of the full Kronecker product.

Define further the *between-class feature second moment*:

$$\boldsymbol{H}^l_{\text{class}} = \underset{c}{\text{Ave}}\, \boldsymbol{h}^l_c \boldsymbol{h}^{l\top}_c,$$

and the *within-class feature covariance*,

$$\boldsymbol{H}^l_{\text{within}} = \underset{i,c}{\text{Ave}} (\boldsymbol{h}^l_{i,c} - \boldsymbol{h}^l_c)(\boldsymbol{h}^l_{i,c} - \boldsymbol{h}^l_c)^\top.$$

Using a mean-variance decomposition, one can show that

$$\boldsymbol{H}^l = \boldsymbol{H}^l_{\text{class}} + \boldsymbol{H}^l_{\text{within}}.$$

Define also a class-specific second moment matrix,

$$\boldsymbol{H}_c = \underset{i}{\text{Ave}}\, \boldsymbol{h}^l_{i,c} \boldsymbol{h}^{l\top}_{i,c},$$

which is related to the global one through the following equation,

$$\boldsymbol{H}^l = \underset{c}{\text{Ave}}\, \boldsymbol{H}^l_c.$$

7.5 C outliers attributable to $\boldsymbol{H}_{\text{class}}$ and bulk attributable to $\boldsymbol{H}_{\text{within}}$

In Figure 7 we attribute via knockouts the top C outliers in the spectrum of $\boldsymbol{H}^l$ to the spectrum of $\boldsymbol{H}^l_{\text{class}}$. One can similarly attribute via knockouts the largest of these C outliers to the second moment of the global mean. Specifically, Figure 7 shows scatter plots of $\lambda_i(\boldsymbol{H}^l)$ versus $\lambda_i(\boldsymbol{H}^l \nparallel \boldsymbol{H}^l_{\text{class}})$. The last column of the second row illustrates the projection of a blue and an orange point on the x- and y-axes. Notice how their projections on the y-axis are dramatically farther apart than their projections on the x-axis. The projections on the y-axis are far apart because the spectrum of $\boldsymbol{H}^l$ has outliers, corresponding to the blue points, which are separated from a bulk, corresponding to the orange points. The same two points are quite close when projected on the x-axis. This is because the outliers are close to the bulk after the knockout procedure. Since the knockout eliminated the outliers, these would-be outliers are attributed to the matrix we knocked out, i.e., $\boldsymbol{H}^l_{\text{class}}$. Roughly speaking, the orange points are situated along the identity line, which means the bulk of the spectrum is unaffected by knockouts. The blue points, on the other had, corresponding to the top-C eigenvalues, deviate substantially from the identity line. In fact, in the last few columns they deviate so strongly that they separate markedly from the orange points; see the fourth and fifth panels of the third row. In short, there is a bulk-and-outliers structure in the spectrum of $\boldsymbol{H}^l$ and the outliers emerge from the bulk with increasing depth.

7.6 Gradual separation and whitening of feature class-means

Figure 8 plots the same spectra of $\boldsymbol{H}^l$, together with the C eigenvalues of $\boldsymbol{H}^l_{\text{class}}$. Compare and contrast the first and last columns of the first row. In the first column of the first row, the top-C eigenvalues associated with the eigenvalues of $\boldsymbol{H}^l_{\text{class}}$ are smaller than the top-C eigenvalues of $\boldsymbol{H}^l$. Those eigenvalues of $\boldsymbol{H}^l_{\text{class}}$ are too small to induce outliers in

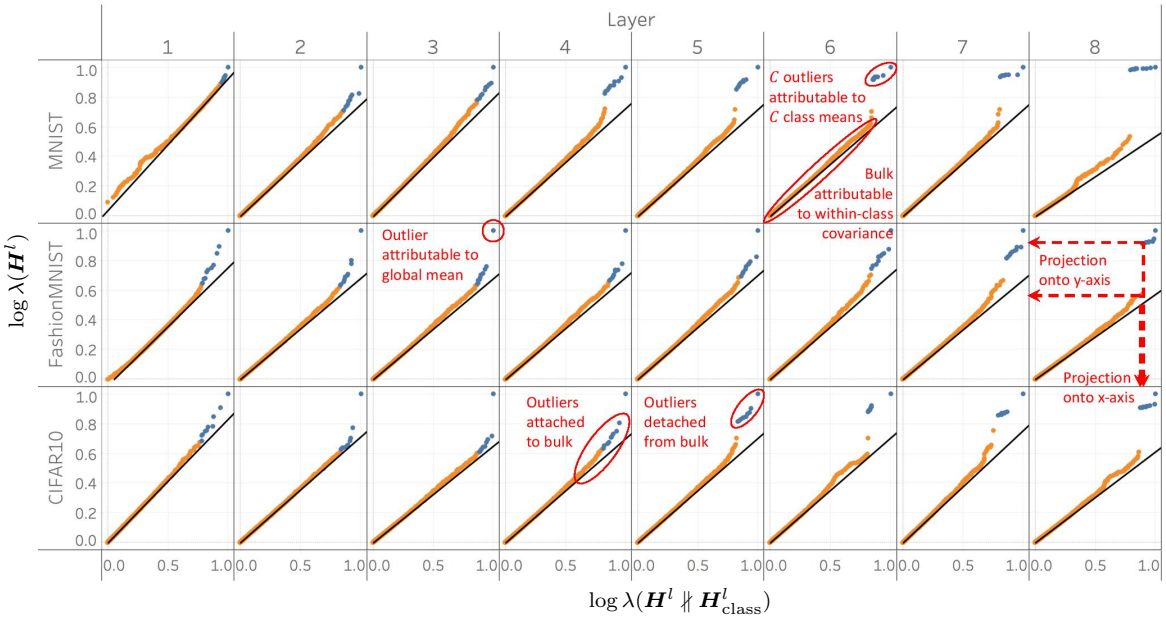


Figure 7: **Attribution via knockouts of spectral features in the spectrum of the features.** Each row of panels corresponds to a different dataset and each column to a different layer. Each panel shows a scatter plot of the top 750 log-eigenvalues of $\boldsymbol{H}^l$ versus those of $\boldsymbol{H}^l \nparallel \boldsymbol{H}^l_{\text{class}}$. The black curve within each panel is the identity line. The top-C outliers are marked in blue and the rest of the eigenvalues are marked in orange. The sixth column of the first row shows the C outliers, attributable to the between-class covariance, $\boldsymbol{H}^l_{\text{class}}$, and the bulk, attributable to the within-class covariance, $\boldsymbol{H}^l_{\text{within}}$. The third column of the second row shows the biggest outlier, which is attributable to the global mean, $\boldsymbol{h}_G$. The fourth and fifth columns of the third row show the layer in which the C outliers separate from the bulk. The network is an eight-layer MLP with 2048 neurons in each hidden layer. For visibility purposes, the values on the x-axis and y-axis are normalized to the range $[0, 1]$. See Section 7.5 for further details.

the spectrum of $\boldsymbol{H}^l_{\text{class}}$. In the last column of the first row, the top-C eigenvalues of $\boldsymbol{H}^l$ and the top-C eigenvalues of $\boldsymbol{H}^l_{\text{class}}$ are close to each other–which is already expected; recall that we attributed these outliers to $\boldsymbol{H}^l_{\text{class}}$ in the previous subsection. These eigenvalues are dramatically larger than later eigenvalues. Again, there is a bulk and C eigenvalues sticking out of the bulk.

Observe the increasing separation, with increasing depth, of the C eigenvalues of $\boldsymbol{H}^l_{\text{class}}$ from the bulk. Underlying this, the class means have larger magnitude with depth and this causes separation. Statistically, this means the features at later layers are more discriminative.

Observe the eigenvalues of $\boldsymbol{H}^l_{\text{class}}$ in different columns of the second row of panels, together with the whiskers highlighting the ratio between the second-largest and smallest eigenvalue. Notice how the ratio decreases as function of depth, i.e., the eigenvalues of $\boldsymbol{H}^l_{\text{class}}$ become increasingly closer to each other. This implies the optimization algorithm finds features

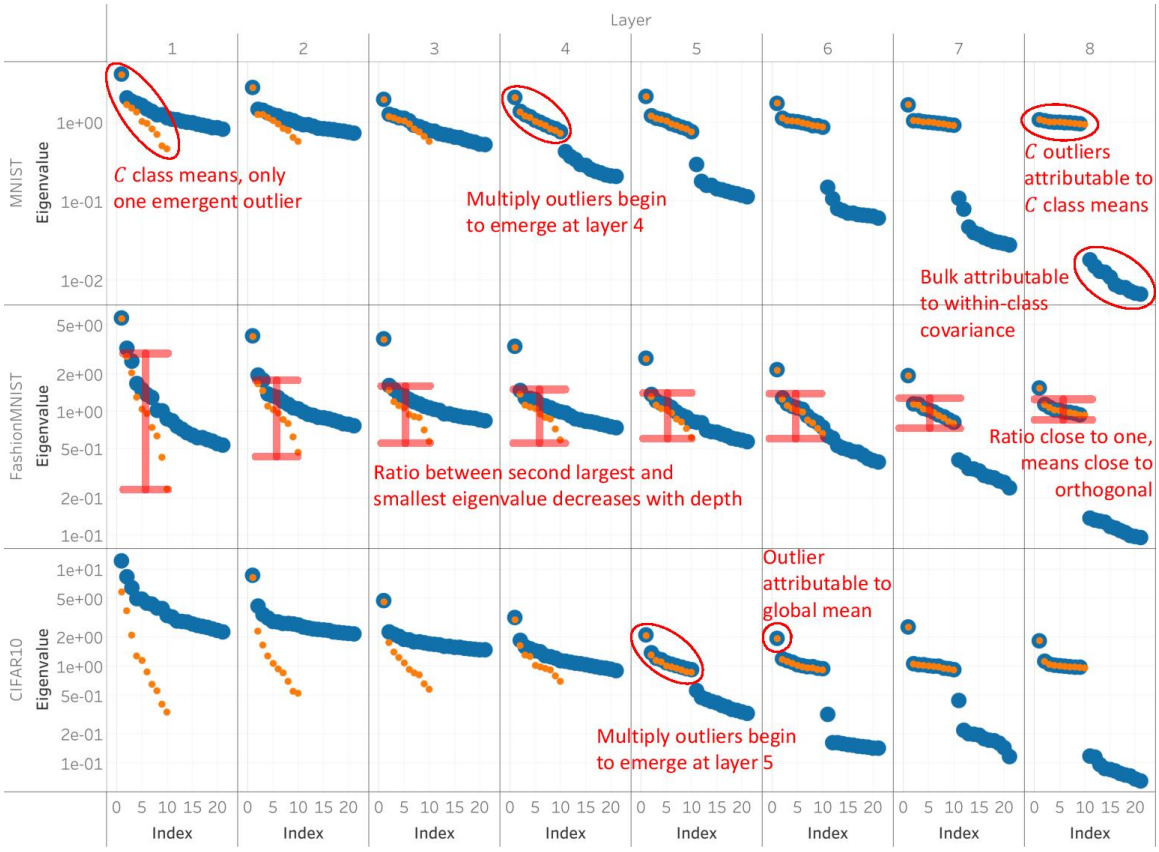


Figure 8: **Scree plot of top eigenvalues in the spectrum of the features.** An alternative presentation of the data in Figure 7. Each row of panels corresponds to a different dataset and each column to a different layer. Each panel depicts spectra of $\boldsymbol{H}^l$ in blue and $\boldsymbol{H}^l_{\text{class}}$ in orange. Eigenvalues in each panel are normalized by the median of $\left\{\log\left(\lambda_i(\boldsymbol{H}^l_{\text{class}})\right)\right\}_{i=1}^{C}$. The panel in the last column of the first row highlights the "bulk and C outliers" pattern. The panels in the second row show whiskers whose edges are located at the second largest and at the smallest eigenvalue of $\boldsymbol{H}^l_{\text{class}}$. Notice how the ratio between the whisker edges decreases as function of depth, until a point where all eigenvalues are of similar magnitude. The third row highlights the top outlier, which is attributable to the global mean, $\boldsymbol{h}_G$. The network is an eight-layer MLP with 2048 neurons in each hidden layer. For more details, see Section 7.6.

whose class means are increasingly closer to orthogonal (once these class means are centered by a global mean, which eliminates the first outlier).

7.7 Cross-class block structure in backpropagated errors

Consider the weighted second moment of backpropagated errors:

$$\boldsymbol{\Delta}^l = \frac{1}{NC} \sum_{i,c,c'} w_{i,c,c'} \boldsymbol{\delta}^l_{i,c,c'} \boldsymbol{\delta}^l_{i,c,c'}{}^\top.$$

Define the *backpropagated errors cross-class means*:

$$\boldsymbol{\delta}^l_{c,c'} = \sum_i \pi_{i,c,c'} \boldsymbol{\delta}^l_{i,c,c'},$$

and their corresponding *within-cross-class covariance*:

$$\boldsymbol{\Delta}^l_{\text{within}} = \sum_{i,c,c'} w_{i,c,c'} (\boldsymbol{\delta}^l_{i,c,c'} - \boldsymbol{\delta}^l_{c,c'})(\boldsymbol{\delta}^l_{i,c,c'} - \boldsymbol{\delta}^l_{c,c'})^\top.$$

These equations group together backpropagated errors for a fixed pair of c, c'. Define also the *backpropagated errors class means*:

$$\boldsymbol{\delta}^l_{c} = \sum_{c' \neq c} \pi_{c,c'} \boldsymbol{\delta}^l_{c,c'},$$

the *between-class backpropagated error second moment*:

$$\boldsymbol{\Delta}^l_{\text{class}} = \sum_c w_c \boldsymbol{\delta}^l_c {\boldsymbol{\delta}^l_c}^\top,$$

and the *between-cross-class backpropagated error covariance*:

$$\boldsymbol{\Delta}^l_{\text{cross}} = \sum_{\substack{c' \neq c \\ c}} w_{c,c'} (\boldsymbol{\delta}^l_{c,c'} - \boldsymbol{\delta}^l_{c'})(\boldsymbol{\delta}^l_{c,c'} - \boldsymbol{\delta}^l_{c'})^\top.$$

In these equations, the backpropagated errors with a fixed c (and possibly varying c') are clustered together. Leveraging these definitions, we obtain the following decomposition of $\boldsymbol{\Delta}$:

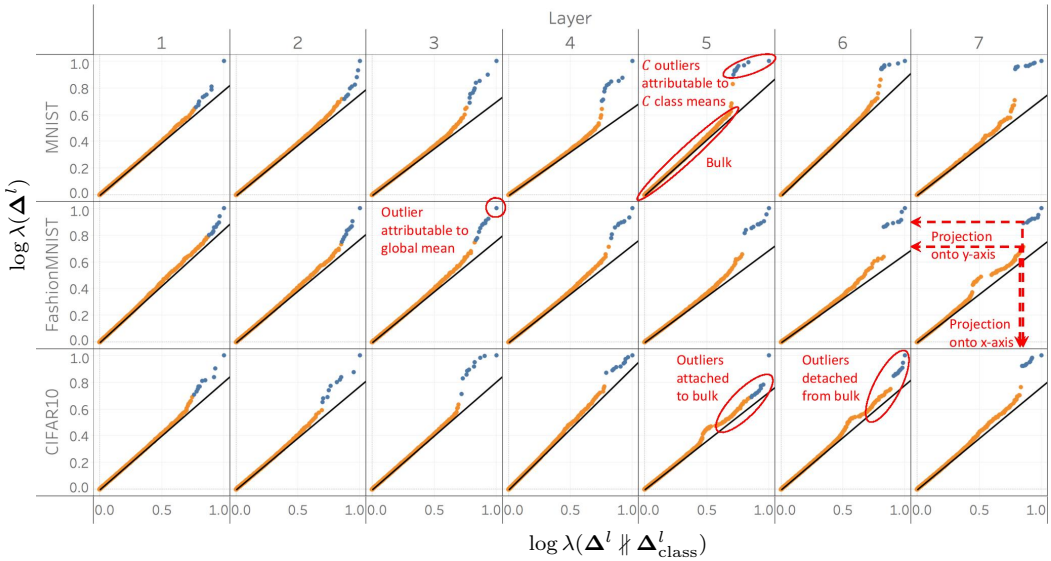
$$\boldsymbol{\Delta}^l = \boldsymbol{\Delta}^l_{\text{class}} + \boldsymbol{\Delta}^l_{\text{cross}} + \boldsymbol{\Delta}^l_{\text{within}} + \underbrace{\sum_c w_{c,c} \boldsymbol{\delta}^l_{c,c} {\boldsymbol{\delta}^l_{c,c}}^\top}_{\boldsymbol{\Delta}_{c=c'}}.$$

The above decomposition is similar to the decomposition of the gradients in Section 6 and its proof is therefore omitted. We can also define a class-specific weighted second moment given by,

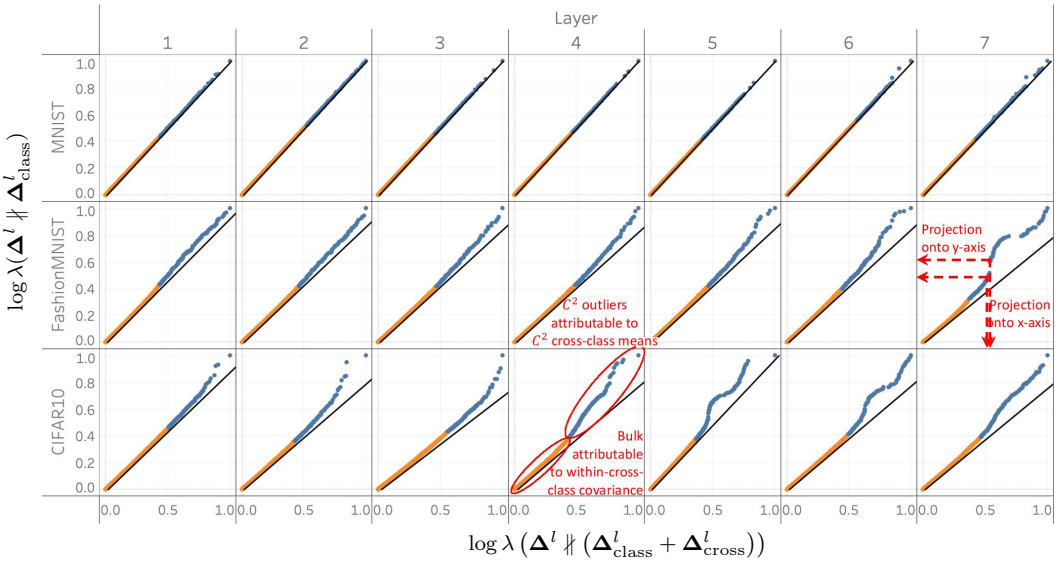
$$\boldsymbol{\Delta}^l_c = \frac{1}{N} \sum_{i,c'} w_{i,c,c'} \boldsymbol{\delta}^l_{i,c,c'} {\boldsymbol{\delta}^l_{i,c,c'}}^\top.$$

7.8 C outliers attributable to $\boldsymbol{\Delta}_{\text{class}}$, C^2 outliers attributable to $\boldsymbol{\Delta}_{\text{cross}}$ and bulk attributable to $\boldsymbol{\Delta}_{\text{within}}$

In Figure 9a we attribute via knockouts the top C outliers in $\boldsymbol{\Delta}^l$ to the spectrum of $\boldsymbol{\Delta}^l_{\text{class}}$. Specifically, Figure depicts scatter plots of $\lambda_i(\boldsymbol{\Delta}^l)$ versus $\lambda_i(\boldsymbol{\Delta}^l \nparallel \boldsymbol{\Delta}^l_{\text{class}})$. The last column of the second row illustrates the projection of a blue and an orange point on the principal axes. Their projections on the y-axis are dramatically farther apart than their projections on the x-axis. Their projections on the y-axis are far apart because the spectrum of $\boldsymbol{\Delta}^l$ has outliers, corresponding to the blue points, which are separated from a bulk, corresponding



(a) **Top-C outliers attributed to C class means.** Each panel shows a scatter plot of top the 750 log-eigenvalues of $\mathbf{\Delta}^l$ versus those of $\mathbf{\Delta}^l \nparallel \mathbf{\Delta}^l_C$. The top-$C$ outliers are marked in blue and the rest of the eigenvalues are marked in orange. The fifth column of the first row shows the C outliers, attributable to the between-class covariance, $\mathbf{\Delta}^l_{\text{class}}$, and a bulk. The third column of the second row shows the biggest outlier, which is attributable to the global mean, $\boldsymbol{\delta}_G$. The fifth and sixth columns of the third row show the layer in which the C outliers separate from the bulk.



(b) **Top-C^2 outliers attributed to C^2 cross-class means.** Each panel shows a scatter plot of the top 750 log-eigenvalues of $\mathbf{\Delta}^l \nparallel \mathbf{\Delta}^l_{\text{class}}$ versus those of $\mathbf{\Delta}^l \nparallel (\mathbf{\Delta}^l_{\text{class}} + \mathbf{\Delta}^l_{\text{cross}})$. The top-$C^2$ outliers are marked in blue and the rest are marked in orange. The fourth column of the third row shows the C^2 outliers, attributable to the between-cross-class covariance, $\mathbf{\Delta}^l_{\text{cross}}$, and a bulk, attributable to the within-cross-class covariance $\mathbf{\Delta}^l_{\text{within}}$.

Figure 9: **Attribution via knockouts of spectral features in the spectrum of back-propagated errors.** Eight-layer MLP with 2048 neurons in each hidden layer. Within each subfigure, each row of panels corresponds to a different dataset and each column to a different layer. The black curve within each panel is the identity line. For visibility purposes, the values on the x-axis and y-axis are normalized to the range $[0, 1]$. See Section 7.8 for further details.

to the orange points. The same two points are quite close when projected on the x-axis. This is because the outliers are close to the bulk after the knockout procedure. Since the knockout eliminated the outliers, these would-be outliers are attributed to the matrix we knocked out, i.e., $\mathbf{\Delta}^l_{\text{class}}$. The orange points are situated along the identity line, which means the bulk of the spectrum is unaffected by knockouts. The blue points, corresponding to the top-C eigenvalues, deviate substantially from the identity line. In some panels they deviate so strongly that they separate markedly from the orange points; see fifth and sixth panels of the third row. In short, there is a bulk-and-outliers structure in the spectrum of $\mathbf{\Delta}^l$ and the outliers emerge from the bulk with increasing depth.

In Figure 9b we attribute via knockouts the top C^2 outliers in the spectrum of $\mathbf{\Delta}^l$ to the spectrum of $\mathbf{\Delta}_{\text{class}} + \mathbf{\Delta}_{\text{cross}}$. Specifically, Figure 9b depicts scatter plots of $\lambda_i(\mathbf{\Delta}^l \nVdash \mathbf{\Delta}^l_{\text{class}})$ versus $\lambda_i(\mathbf{\Delta}^l \nVdash (\mathbf{\Delta}^l_{\text{class}} + \mathbf{\Delta}_{\text{cross}}))$. The seventh column of the second row illustrates the projection of two blue points–not a blue and an orange point, as was done previously–onto the principal axes. Notice the gap dividing the set of all blue points into two groups: one that is close to the orange points, and another which is well-separated from the orange points. Notice how the projections of these two blue points onto the y-axis are dramatically father apart than their projections on the x-axis. Their projection onto the y-axis are far apart because the spectrum of $\mathbf{\Delta}^l \nVdash \mathbf{\Delta}^l_{\text{class}}$ has outliers, corresponding to one of the groups of blue points. These are separated from a bulk, comprised of the other group of blue points and also the orange points. The same two points are quite close when projected on the x-axis. This is because the outliers are close to the bulk after the knockout procedure. Since knocking out $\mathbf{\Delta}_{\text{cross}}$ eliminated the outliers in $\mathbf{\Delta}^l \nVdash (\mathbf{\Delta}^l_{\text{class}} + \mathbf{\Delta}_{\text{cross}})$, these would-be outliers are attributable to the matrix we knocked out, i.e., $\mathbf{\Delta}^l_{\text{cross}}$. Some of the blue points and all of the orange points are situated along the identity line, which means the bulk of the spectrum is unaffected by the knockout procedure.

7.9 Gradual separation and whitening of backpropagated error class-means

Figure 10 plots the same spectra of $\mathbf{\Delta}^l$, together with the eigenvalues of $\mathbf{\Delta}^l_{\text{class}}$ and $\mathbf{\Delta}^l_{\text{class}} + \mathbf{\Delta}^l_{\text{cross}}$. Compare and contrast the first, fifth and seventh column of the second row. In the first column of the second row, the top-C eigenvalues associated with the eigenvalues of $\mathbf{\Delta}^l_{\text{class}}$, and the next C^2 outliers associated with the eigenvalues of $\mathbf{\Delta}^l_{\text{cross}}$, are smaller than the top eigenvalues of $\mathbf{\Delta}^l$. Those eigenvalues of $\mathbf{\Delta}^l_{\text{class}}$ and $\mathbf{\Delta}^l_{\text{cross}}$ are too small to induce outliers in the spectrum of $\mathbf{\Delta}^l$. In the fifth column of the second row, the top-C eigenvalues of $\mathbf{\Delta}^l$ and the top-C eigenvalues of $\mathbf{\Delta}^l_{\text{class}}$ are close to each other, which is already expected; recall that we attributed the top-C outliers in the spectrum of $\mathbf{\Delta}^l$ to $\mathbf{\Delta}^l_{\text{class}}$ in the previous subsection. Moreover, the C^2 eigenvalues associated with $\mathbf{\Delta}^l_{\text{cross}}$ are smaller than the top C^2 eigenvalues of $\mathbf{\Delta}^l$. Those eigenvalues are too small to induce outliers in the spectrum of $\mathbf{\Delta}^l$. In the seventh column of the second row, some of the top-C^2 eigenvalues of $\mathbf{\Delta}^l$ and the C^2 eigenvalues of $\mathbf{\Delta}^l_{\text{cross}}$ are close to each other. This is again already expected; recall that we attributed the top-C outliers in $\mathbf{\Delta}^l$ to $\mathbf{\Delta}^l_{\text{class}}$ and the top-C^2 outliers in $\mathbf{\Delta}^l$ to $\mathbf{\Delta}^l_{\text{cross}}$.

Observe the increasing separation, with increasing depth, of the C eigenvalues of $\mathbf{\Delta}^l_{\text{class}}$ from the bulk. Underlying this, the class means have larger magnitude with depth and this causes separation. Heuristically, this figure implies that the backpropagated errors

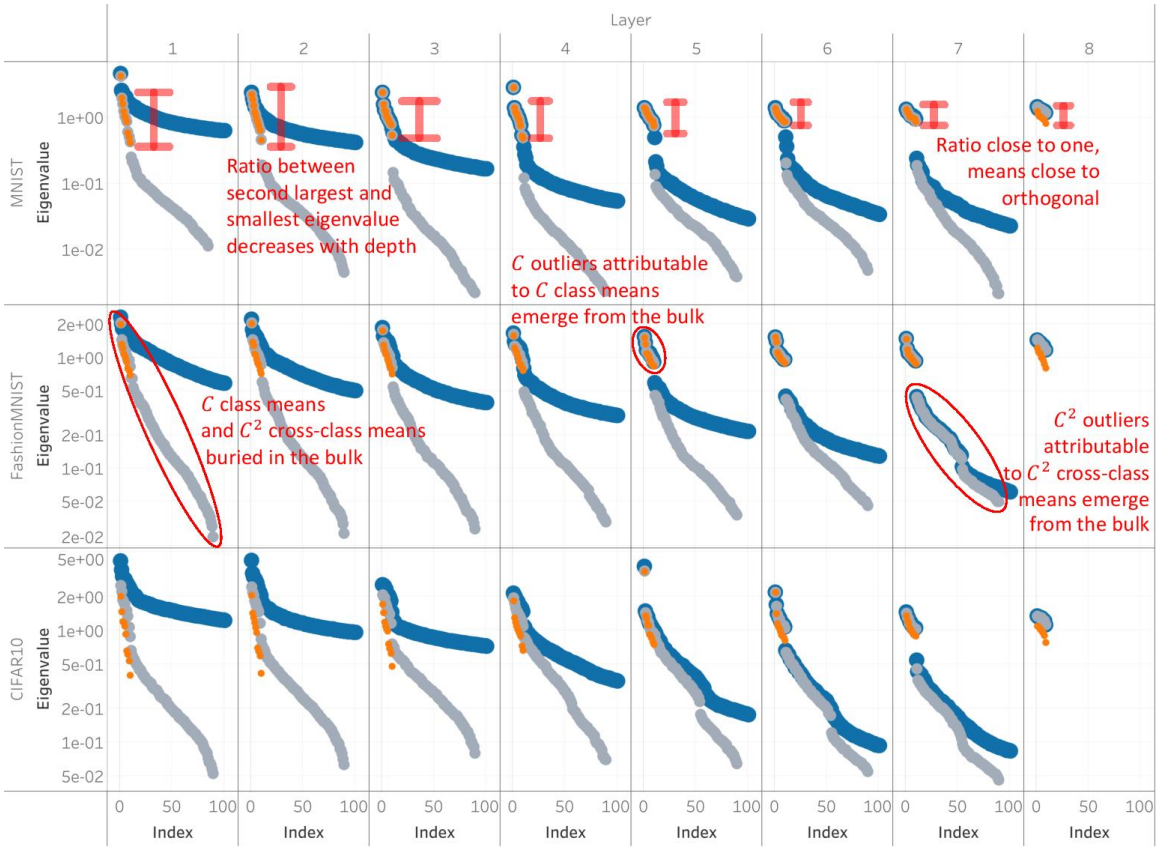


Figure 10: **Scree plot of top eigenvalues in spectra of the backpropagated errors.** An alternative presentation of the data in Figure 9. Each row of panels corresponds to a different dataset and each column to a different layer. Each panel depicts spectra of $\mathbf{\Delta}^l$ in blue, $\mathbf{\Delta}^l_{\text{class}}$ in orange and $\mathbf{\Delta}^l_{\text{class}} + \mathbf{\Delta}^l_{\text{cross}}$ in gray. Eigenvalues in each panel are normalized by the median of $\{\log\big(\lambda_i(\mathbf{\Delta}^l_{\text{class}})\big)\}_{i=1}^{C}$. The first, fifth and eighth columns of the second row track: (i) the emergence of the C outliers from the bulk, (ii) the emergence of the C^2 outliers from the bulk. The panels in the first row show whiskers whose edges are located at the second largest and at the smallest eigenvalue of $\mathbf{\Delta}^l_{\text{class}}$. Notice how the ratio between the whisker edges decreases as function of depth, until a point where all eigenvalues are of similar magnitude. The network is an eight-layer MLP with 2048 neurons in each hidden layer. For more details, see Section 7.9.

contain strong class information in deeper layers, but this information is gradually lost during backpropagation to successively shallower layers.

Observe in the panels of the first row the eigenvalues of $\mathbf{\Delta}^l_{\text{class}}$ throughout the layers. The whiskers highlight the ratio between the second-largest and the smallest eigenvalue. Note how this ratio decreases with depth; by the last layer, all eigenvalues are of fairly similar magnitude. This implies the error class means are ultimately very close to being orthogonal, but their orthogonality is gradually deteriorating as they are backpropagated throughout the layers.

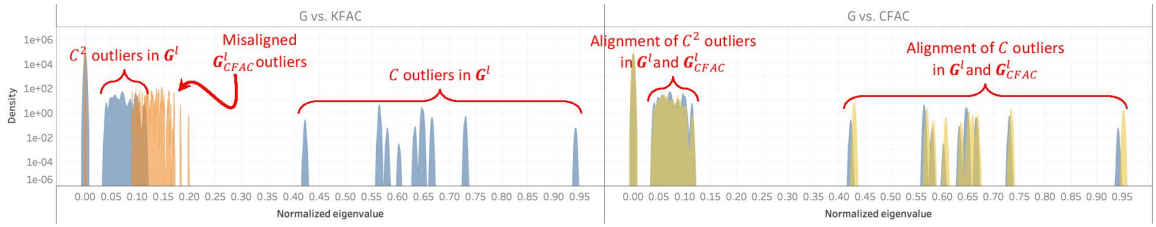


Figure 11: **Spectra of $\boldsymbol{G}^l$, $\boldsymbol{G}^l_{\mathbf{KFAC}}$ and $\boldsymbol{G}^l_{\mathbf{CFAC}}$ for layer $l = 8$.** Each panel plots the spectrum of $\boldsymbol{G}^l$ in blue. The left panel highlights features in the spectrum of $\boldsymbol{G}$, which include C outliers, as well as a mini-bulk composed of C^2 outliers. The main bulk in the spectrum of the last layer features is concentrated and appears as a narrow bump around the origin. In addition, the left panel plots the spectrum of $\boldsymbol{G}^l_{\text{KFAC}}$ in orange, while the right one plots the spectrum of $\boldsymbol{G}^l_{\text{CFAC}}$ in yellow. Notice that outliers in the spectrum of $\boldsymbol{G}^l_{\text{KFAC}}$ **do not** align well with outliers in the spectrum of $\boldsymbol{G}^l$. On the other hand, notice that the C and C^2 outliers in the spectrum of $\boldsymbol{G}^l_{\text{CFAC}}$ **do align** well with corresponding outliers in the spectrum of $\boldsymbol{G}^l$. The network is an eight-layer MLP with 2048 neurons in each hidden layer trained on MNIST. The y-axis is on a logarithmic scale.

7.10 Class-distinct factorized approximate curvature

Traditional quadratic optimization can be readily solved using a single Newton's method step, by moving in the direction of the gradient preconditioned by Hess^{-1}. For nonlinear least squares problems, one can utilize the Gauss–Newton algorithm, which replaces Hess^{-1} by $\boldsymbol{G}^{-1}$ and performs several steps, instead of just one. In the context of deep neural networks, Gauss-Newton is problematic, since the matrix of $\boldsymbol{G}$ is too large for it to be stored in memory. Researchers would like to find other matrices that might precondition the gradient (Grosse and Salakhudinov, 2015; Ye et al., 2018; Wang and Zhang, 2019; Xu et al., 2019; Kylasa et al., 2019; Xu et al., 2020). For example, Martens and Grosse (2015) proposed the KFAC matrix as a proxy for $\boldsymbol{G}$.

Definition 7.1 (KFAC; Martens and Grosse (2015)). *The KFAC matrix is defined as follows:*

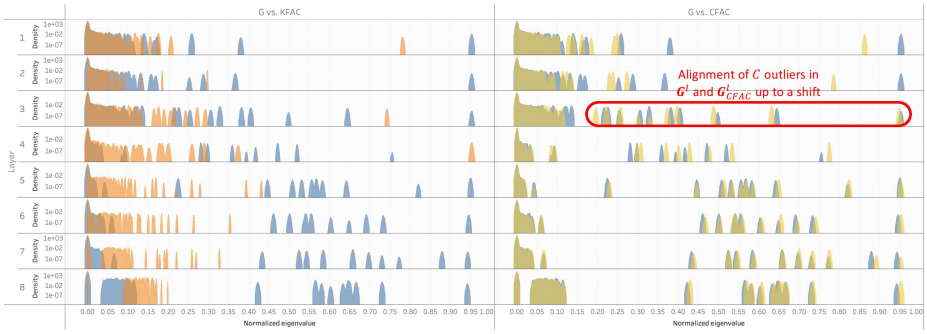
$$\begin{aligned}\boldsymbol{G}_{KFAC} &= \underset{i,c}{\text{Ave}} \left\{ \boldsymbol{h}_{i,c} \boldsymbol{h}_{i,c}^\top \right\} \odot \left(\sum_{i,c,c'} w_{i,c,c'} \boldsymbol{\delta}_{i,c,c'} \boldsymbol{\delta}_{i,c,c'}^\top \right) \\ &= \boldsymbol{H} \odot \boldsymbol{\Delta},\end{aligned}$$

where $\odot$ denotes Khatri-Rao product. The block diagonal KFAC matrix is given by:

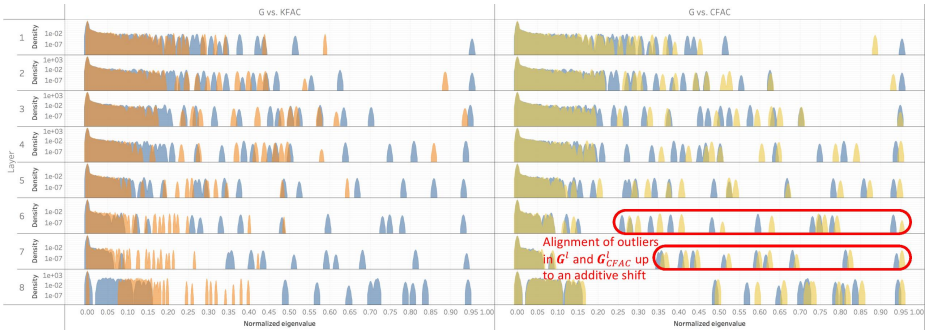
$$\begin{aligned}\boldsymbol{G}^l_{KFAC} &= \underset{i,c}{\text{Ave}} \left\{ \boldsymbol{h}^l_{i,c} {\boldsymbol{h}^l_{i,c}}^\top \right\} \otimes \left(\sum_{i,c,c'} w_{i,c,c'} \boldsymbol{\delta}^l_{i,c,c'} {\boldsymbol{\delta}^l_{i,c,c'}}^\top \right) \\ &= \boldsymbol{H}^l \otimes \boldsymbol{\Delta}^l,\end{aligned}$$

where $\otimes$ is the Kronecker product[6].

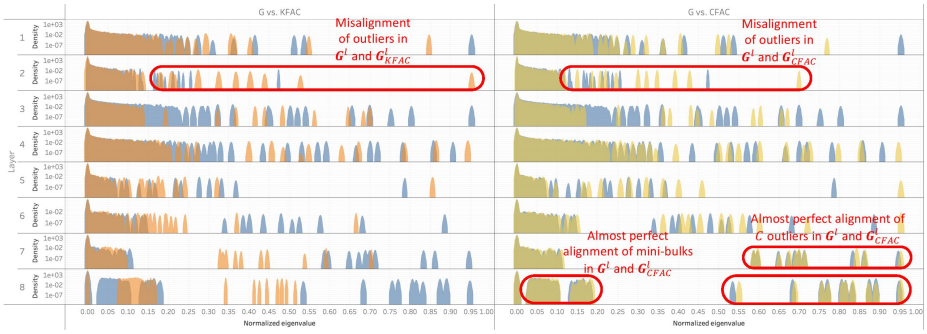
6. Hence the name of their method–Kronecker-Factored Approximation of Curvature (KFAC).



(a) MNIST



(b) FashionMNIST



(c) CIFAR10

Figure 12: **Spectra of G^l, G^l_{KFAC} and G^l_{CFAC}.** Each subfigure corresponds to one specific dataset. Each row of panels corresponds to one specific layer. In each panel, the spectrum of G^l is plotted in blue. The first column of panels plots the spectrum of G^l_{KFAC} in orange, while the second column of panels plots the spectrum of G_{CFAC} in yellow. Notice how the spectra of G^l and G^l_{KFAC} do not align well, as evident by the lack of agreement of blue and orange outliers, mini-bulks and main bulks. Conversely, notice how the spectra of G^l and G^l_{CFAC} align much better, especially at deeper layers. At the seventh and eighth layers of CIFAR10, the outliers and mini-bulks in the spectra of G^l and G^l_{CFAC} align almost perfectly. In other cases, such as the sixth and seventh layers of FashionMNIST, the outliers in the spectra of G^l and G^l_{CFAC} align up to a small shift. In some shallower layers, such as the third layer of MNIST, the outliers in the spectrum of G^l align with those in the spectrum of G^l_{CFAC}. In other cases, such as the second layer of CIFAR10, the misalignment of the outliers in the spectra of G^l and G^l_{CFAC} is as bad as the misalignment between the outliers of G^l and G^l_{KFAC}. The network is an eight-layer MLP with 2048 neurons in each hidden layer. The y-axis is on a logarithmic scale.

The idea of using Kronecker-based preconditioners can be traced to earlier works by Van Loan and Pitsianis (1993), and the use of a block-diagonal approximation can be traced back to Collobert and Bengio (2004). KFAC has proven to be very useful in practice, and since its conception several generalizations and applications were proposed for it (Grosse and Martens, 2016; Ba et al., 2016; Wu et al., 2017b,a; Wang et al., 2019).

Tools developed in this present work can be used to study the spectra of these matrices, and thereby to also understand the extent (or not) of agreement between the spectra of $\boldsymbol{G}$ and $\boldsymbol{G}_{\text{KFAC}}$. When we apply our attribution tools (see left panels of Figures 11 and 12), we notice very distinct failures of approximation. $\boldsymbol{G}_{\text{KFAC}}$ has its mini-bulks and outliers drastically misaligned from those of $\boldsymbol{G}$. We propose an alternative approximation for $\boldsymbol{G}$.

Definition 7.2 (**Class-distinct Factorized Approximate Curvature**). *The CFAC matrix is given by:*

$$\boldsymbol{G}_{CFAC} = \underset{c}{\text{Ave}}\, \boldsymbol{H}_c \odot \boldsymbol{\Delta}_c,$$

and the block diagonal CFAC matrix is given by:

$$\boldsymbol{G}^l_{CFAC} = \underset{c}{\text{Ave}}\, \boldsymbol{H}^l_c \otimes \boldsymbol{\Delta}^l_c.$$

7.11 CFAC is a better proxy for G than KFAC

In Figure 11 we compare the spectra of $\boldsymbol{G}^l$, $\boldsymbol{G}^l_{\text{KFAC}}$ and $\boldsymbol{G}^l_{\text{CFAC}}$ for the last layer of an MLP trained on MNIST. Inspection reveals that the spectra of $\boldsymbol{G}^l_{\text{CFAC}}$ and $\boldsymbol{G}^l$ align much better than do those of $\boldsymbol{G}^l_{\text{KFAC}}$ and $\boldsymbol{G}^l$. In Figure 12, we perform a more comprehensive comparison, which considers all the layers of the MLP across three canonical datasets. At deeper layers, the spectra of $\boldsymbol{G}^l_{\text{CFAC}}$ and $\boldsymbol{G}^l$ again align much better than $\boldsymbol{G}_{\text{KFAC}}$ and $\boldsymbol{G}^l$. At shallower layers, improvement in alignment depends on the dataset.

7.12 Relating the class/cross-class structure in the features, backpropagated errors and G

Recall that $\boldsymbol{G}$, $\boldsymbol{H}$ and $\boldsymbol{\Delta}$ all have C-dimensional eigenspaces attributable to the class-specific mean structure. In KFAC, each of the C features class means would be Khatri-Rao-multiplied by each of the C error class means. In other words, the feature mean of one class would be multiplied by the error mean of another class. In CFAC, on the other hand, the feature mean of a class would **only** be multiplied by the error mean of that class.

The previous subsection suggests that the class means of gradients, which cause the outliers in $\boldsymbol{G}$, can be approximated as the Khatri-Rao product of the C feature class means and the C error class means, i.e.,

$$\boldsymbol{g}_c \approx \boldsymbol{\delta}_c \odot \boldsymbol{h}_c.$$

Moreover, the C^2 cross-class means causing the other visible outliers can be approximated as the Khatri-Rao product of the C feature class means and the C^2 error cross-class means, i.e.,

$$\boldsymbol{g}_{c,c'} \approx \boldsymbol{\delta}_{c,c'} \odot \boldsymbol{h}_c.$$

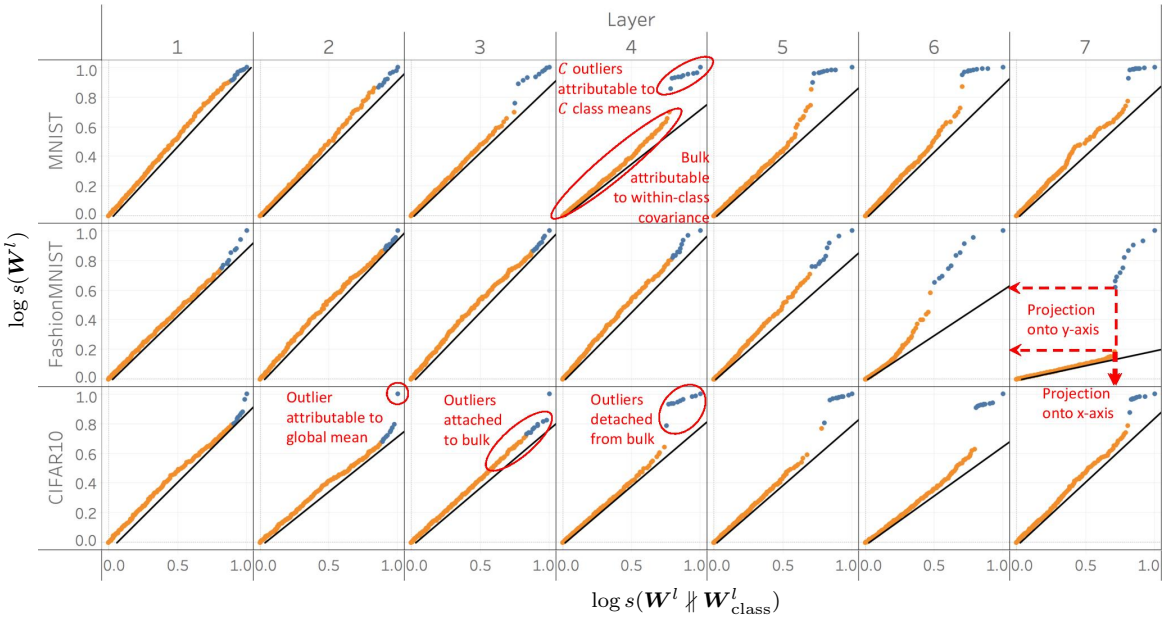


Figure 13: **Attribution via knockouts of spectral features in the spectrum of the weights.** Each row of panels corresponds to a specific dataset and each column to a specific layer. Each panel shows a scatter plot of the top 150 log singular values of $\boldsymbol{W}^l$ versus those of $\boldsymbol{W}^l \nparallel \boldsymbol{W}^l_{\text{class}}$. The black curve within each panel is the identity line. The top-C outliers are marked in blue; other singular values are marked in orange. The fourth column of the first row shows the C outliers, attributable to the between-class covariance, $\boldsymbol{W}^l_{\text{class}}$, and the bulk, attributable to the within-class covariance, $\boldsymbol{W}^l_{\text{within}}$. The second column of the third row shows the biggest outlier, which is attributable to the global mean, $\boldsymbol{\delta}_G \boldsymbol{h}_G^\top$. The third and fourth columns of the third row show the layer in which the C outliers separate from the bulk. The network is an eight-layer MLP with 2048 neurons in each hidden layer. For plotting purposes, values on the x-axis and y-axis are normalized to the range $[0, 1]$. See Section 7.13 for further details.

7.13 Relating the class structure in features, backpropagated errors and weights

In this subsection we assume a weight decay regularization is used to train the network. Recalling Equation (7.1), the gradient of the loss with respect to the l'th layer weight matrix $\boldsymbol{W}^l$ is given by

$$\frac{\partial \mathcal{L}(\theta)}{\partial \boldsymbol{W}^l} = \operatorname*{Ave}_{i,c} \frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_c)}{\partial \boldsymbol{W}^l} = \operatorname*{Ave}_{i,c} \boldsymbol{\delta}^l_{i,c,c} \boldsymbol{h}^{l-1\top}_{i,c} + \eta \boldsymbol{W}^l.$$

Setting the above gradient to zero, we obtain

$$\boldsymbol{W}^l = -\frac{1}{\eta} \operatorname*{Ave}_{i,c} \boldsymbol{\delta}^l_{i,c,c} \boldsymbol{h}^{l-1\top}_{i,c}.$$

Define the *between-class weight matrix*

$$\boldsymbol{W}^l_{\text{class}} = -\frac{1}{\eta} \operatorname*{Ave}_c \boldsymbol{\delta}^l_{c,c} \boldsymbol{h}_c^{l-1\top},$$

where

$$\boldsymbol{\delta}^l_{c,c} = \operatorname*{Ave}_i \boldsymbol{\delta}^l_{i,c}.$$

In Figure 13 we attribute via knockouts the top C outliers in the spectrum of $\boldsymbol{W}^l$ to the spectrum of $\boldsymbol{W}^l_{\text{class}}$. Similarly we could attribute via knockouts the largest of these C outliers to the second moment of the global mean. Specifically, Figure 7 shows scatter plots of singular values of $\boldsymbol{W}^l$ versus those of $\boldsymbol{W}^l \nparallel \boldsymbol{W}^l_{\text{class}}$ (here we use the definition of projection knockouts for non-square matrices). The last column of the second row illustrates the projection of a blue and an orange point on the x- and y-axes. The projections on the y-axis are well-separated, which implies the spectrum of $\boldsymbol{W}^l$ has C outliers, corresponding to the blue points, which are separated from a bulk, corresponding to the orange points. The projections onto the x-axis are not well-separated, which implies that the outliers if any, are largely eliminated after the knockout procedure. Since the knockout eliminated the outliers, these would-be outliers are attributed to the matrix we knocked out, i.e., $\boldsymbol{W}^l_{\text{class}}$. Notice how the orange points are situated along the identity line, which means the bulk of the spectrum is unaffected by the knockout procedure, whereas the blue points, corresponding to the top-C singular values, deviate substantially from the identity line. In fact, in the last few columns they deviate so much that they separate from the orange points, as shown by the third and fourth panels of the third row. This implies a 'bulk-and-outliers' pattern exists in the spectrum of $\boldsymbol{W}^l$, and the outliers emerge from the bulk as we move towards deeper layers.

8. Multinomial logistic regression

The emergence of the pattern in Figure 1 is not restricted to spectra originating from deep neural networks, as it already appears in simpler examples. In this section, we study the spectrum of the FIM of a multinomial logistic regression classifier trained on Gaussian Mixture Model data in order to obtain intuitive understanding of (i) the different components in the spectrum of $\boldsymbol{G}$; and (ii) the relation between these components and the misclassification error.

Since the top layer of a deep network is equivalent to a multinomial logistic regression model, one could think of this section as studying the spectrum of the partial FIM associated with just the top layer, assuming that all lower layers are fixed and the last-layer features are distributed as a Gaussian Mixture Model.

8.1 FIM of multinomial logistic regression

Assuming our network is a multinomial logistic regression classifier, we have

$$f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}) = \boldsymbol{W}\boldsymbol{x}_{i,c}, \quad \boldsymbol{\theta} = \operatorname{vec}(\boldsymbol{W}),$$

where we denoted by $\operatorname{vec}(\cdot)$ the operator forming a vector from a matrix by stacking its columns as subvector blocks within a single vector. Denote by $\otimes$ the Kronecker product.

Using the property of Kronecker products, $\text{vec}(\boldsymbol{AB}) = (\boldsymbol{B}^\top \otimes \boldsymbol{I}_C)\,\text{vec}(\boldsymbol{A})$, we can rewrite the classifier as follows:

$$f(\boldsymbol{x}_{i,c};\boldsymbol{\theta}) = \boldsymbol{W}\boldsymbol{x}_{i,c} = (\boldsymbol{x}_{i,c}^\top \otimes \boldsymbol{I}_C)\,\text{vec}(\boldsymbol{W}) = (\boldsymbol{x}_{i,c}^\top \otimes \boldsymbol{I}_C)\boldsymbol{\theta}.$$

The above implies that

$$\frac{\partial f(\boldsymbol{x}_{i,c};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}^\top = \boldsymbol{x}_{i,c} \otimes \boldsymbol{I}_C,$$

and also

$$\left.\frac{\partial^2 \ell(\boldsymbol{z}, \boldsymbol{y}_c)}{\partial \boldsymbol{z}^2}\right|_{\boldsymbol{z}_{i,c}} = \text{diag}(\boldsymbol{p}_{i,c}) - \boldsymbol{p}_{i,c}\boldsymbol{p}_{i,c}^\top,$$

where $\boldsymbol{p}_{i,c} \in \mathbb{R}^C$ denotes the predicted probability vector of the i'th example belonging to the c'th class. The above identity was proven by Böhning (1992). Plugging these into the definition of $\boldsymbol{G}$ in Equation (5.1), we obtain

$$\boldsymbol{G} = \underset{i,c}{\text{Ave}} \left\{ (\boldsymbol{x}_{i,c} \otimes \boldsymbol{I}_C)(\text{diag}(\boldsymbol{p}_{i,c}) - \boldsymbol{p}_{i,c}\boldsymbol{p}_{i,c}^\top)(\boldsymbol{x}_{i,c} \otimes \boldsymbol{I}_C)^\top \right\}.$$

8.2 Spectrum of FIM under generative model setting

Definition 8.1 (Canonical classification model). *An instance of the canonical classification model $\mathcal{CCM}(D, C)$ consists of the following components:*

- *C canonical vector class means, $\boldsymbol{e}_c \in \mathbb{R}^D$ i.e.,*

$$\boldsymbol{e}_c = [0, \ldots, 0, \underbrace{1}_{index\ c}, 0, \ldots, 0].$$

- *NC independently normally distributed D-dimensional sample deviations:*

$$\boldsymbol{z}_{i,c} \sim \mathcal{N}(0, \boldsymbol{I}), \quad 1 \leq c \leq C, \quad 1 \leq i \leq N.$$

- *A scalar representing the signal-to-noise ratio t.*

The above give rise to a set $\{\boldsymbol{x}_{i,c}\}_{i,c}$ obeying the class block structure, whose elements satisfy:

$$\boldsymbol{x}_{i,c} = t\ \boldsymbol{e}_c + \boldsymbol{z}_{i,c}.$$

As a side note, since the distribution of the sample deviations is rationally invariant, the proofs of following claims could be easily extended to a more general case, whereby the canonical class vector means are rotations of the ones defined above. We can further simplify the $\mathcal{CCM}$ model by imposing the following assumption.

Definition 8.2 (Symmetric probabilities). *The probabilities predicted by the multinomial logistic regression classifier are symmetric if the following holds:*

$$p_{i,c,c'} = \begin{cases} 1-\alpha, & \text{for } c = c' \\ \frac{\alpha}{C-1}, & \text{for } c \neq c' \end{cases},$$

for some constant α satisfying $\frac{1}{C} \leq 1-\alpha \leq 1$, where $p_{i,c,c'}$ is the probability of the i'th example in the c'th class belonging to cross-class c'. In other words, the probabilities are independent of the sample index i and the cross-class c', assuming $c' \neq c$. The vector of probabilities will be denoted by $\boldsymbol{p}_{\cdot,c}$. The correct classification rate $1-\alpha$ is lower bounded by $\frac{1}{C}$ – the rate obtained by randomly guessing a label out of C possibilities.

The following lemma, whose proof is deferred to Appendix B, proves that in the canonical classification model the expected FIM admits a particularly simple form.

Lemma 8.1 (Expected FIM of multinomial logistic regression trained on $\mathcal{CCM}$). *The expected FIM of multinomial logistic regression, with symmetric probabilities, trained on $\mathcal{CCM}(D,C)$, is given by:*

$$\mathbb{E}\,\boldsymbol{G} = \frac{s}{C} \cdot \text{blkdiag}(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_C, \boldsymbol{0}_{DC-C^2}) + \boldsymbol{I} \otimes \bar{\boldsymbol{U}},$$

where

$$\begin{aligned} s &= t^2 \\ \boldsymbol{U}_c &= \text{diag}(\boldsymbol{p}_{\cdot,c}) - \boldsymbol{p}_{\cdot,c}\boldsymbol{p}_{\cdot,c}^\top, \\ \bar{\boldsymbol{U}} &= \operatorname*{Ave}_c \boldsymbol{U}_c, \end{aligned}$$

and the operator blkdiag$(\cdot)$ *forms a block diagonal matrix with $\boldsymbol{U}_c$ in its c'th block and zeros elsewhere.*

Notice that the $C \times C$ matrices $\boldsymbol{U}_c$ are in general of rank C and the block diagonal matrix is of rank C^2. Hence, the expected FIM is a perturbation of a rank-C^2 matrix. The following theorem, proven in Appendix B, describes the spectrum of this matrix.

Theorem 8.1 (Spectrum of expected FIM of multinomial logistic regression trained on $\mathcal{CCM}$). *The spectrum of the expected FIM of multinomial logistic regression, with symmetric probabilities, trained on $\mathcal{CCM}(D,C)$, is given by:*

$$\lambda_i\left(\mathbb{E}\,\boldsymbol{G}\right) = \begin{cases} \frac{\alpha}{C-1}\left(s(1-\alpha) + \left(2 - \alpha\frac{C}{C-1}\right)\right), & \text{for } 1 \leq i \leq C \\ \frac{\alpha}{C-1}\left(\frac{s}{C} + \left(2 - \alpha\frac{C}{C-1}\right)\right), & \text{for } C < i \leq C(C-1) \\ \frac{\alpha}{C-1}\left(2 - \alpha\frac{C}{C-1}\right), & \text{for } C(C-1) < i \leq D(C-1) \\ 0, & \text{for } D(C-1) < i \leq DC \end{cases}.$$

Notice how the spectrum has C outliers, a mini-bulk consisting of approximately C^2 eigenvalues, and a main bulk.

The expression $\frac{\alpha}{C-1}\left(2-\alpha\frac{C}{C-1}\right)$ appears in all eigenvalues and is therefore irrelevant when comparing their relative magnitudes. It is clear that the mini-bulk is bigger than the main bulk since its eigenvalues have the additional non-negative term $\frac{\alpha}{C-1}\frac{s}{C}$. Moreover, since we assumed $1-\alpha > \frac{1}{C}$, we have that $\frac{\alpha}{C-1}s(1-\alpha) > \frac{\alpha}{C-1}\frac{s}{C}$, which implies the top-$C$ outliers are bigger than the mini-bulk.

Notice how increasing the squared signal-to-noise ratio s results in: (i) the separation of the top-C outliers from the mini-bulk (because $1-\alpha$ would increase); (ii) the separation of the top-C outliers from the bulk; and (iii) the separation of the mini-bulk from the main bulk. As such, the ratio of outliers to mini-bulk, the ratio of outliers to bulk, and the ratio of mini-bulk to bulk are all predictive of misclassification.

9. Overview of literature in light of this paper

In light of the insights gathered throughout this work, we can now view in a different light several papers in the literature.

9.1 Start looking at layer-wise features and backpropagated errors, stop focusing on the Hessian

In the last year, a large amount of research effort has been devoted into investigating the spectrum of the Hessian. This paper shows that the Hessian is the most complicated quantity one could possibly investigate since: (a) it inherits its cross-class structure from the product of features and backpropagated errors; (b) its cross-class structure, caused by the FIM, is perturbed by another matrix $\boldsymbol{E}$; and (c) it blends the information across the layers into a single quantity. The research community should therefore move from studying the Hessian into studying the layer-wise features and backpropagated errors.

Zhang et al. (2019) raised similar concerns, stating that the analysis of the optimization landscape would be better performed through the study of individual layers, rather than a single holistic quantity, such as the Hessian. Jiang et al. (2018) further motivated the investigation of layer-wise features by showing how generalization error can be predicted from margins of layer-wise features.

9.2 Start looking at class/cross-class means and covariances, stop looking at eigenvalues

The most surprising contribution of this paper is that it shows the inadequacy of eigenanalysis in revealing the fundamental structure in deep learning. It is true the spectrum reflects this structure, as seen through the plethora of measurements made in the literature, but the spectrum does not explain this structure, the class and cross-class means do.

Throughout this paper we consider the setting in which the number of classes C is smaller than the dimension of features in a certain layer. However, if C is bigger than the dimension of the features, then the spectrum would not manifest any bulk-and-outliers structure. However, class means, cross-class means and within-cross-class covariances would still be perfectly valid quantities to measure and study.

Even if the number of classes is small, if the class means are not gigantic, we may see that the spectrum has no outliers. However, the class and cross-class means are still present in the data, they are just not visible through eigenanalysis.

9.3 The effect of batch normalization on the bulk-and-outliers structure

Ghorbani et al. (2019) claim that "batch normalization (Ioffe and Szegedy, 2015) pushes outliers back into the bulk" and "hypothesize a mechanistic explanation for why batch normalization speeds up optimization: it does so via suppression of outlier eigenvalues which slow down optimization".

This paper explains their empirical observation. Specifically, the top-C outliers in the spectrum of the Hessian are caused by a Khatri-Rao product between the feature global mean, $\boldsymbol{h}_G$, and backpropagated errors class means $\boldsymbol{\delta}_c$. The whitening step of batch normalization subtracts the global mean from the features, which significantly decreases their class-mean magnitude. As a result, the outliers in the Hessian are pushed back into the main bulk. Jacot et al. (2019b) provided a similar explanation for why batch normalization accelerates training by proving that the top outlier in the spectrum of the NTK matrix (Jacot et al., 2018) is suppressed as a result of batch normalization.

However, the fact that these outliers are pushed back by batch normalization does not mean they are completely removed, nor does it lessen their fundamental significance. The outliers are caused by class means, which separate from the bulk as function of depth. They represent the separation of class information from noise and are of utmost importance in studying the classification performance of deep neural networks.

9.4 Flatness conjecture

Hochreiter and Schmidhuber (1997) conjectured that the flatness of the loss function around the minima found by SGD is correlated with good generalization. Recent empirical work by Keskar et al. (2016); Jastrzębski et al. (2017, 2018a); Jiang et al. (2019); Lewkowycz et al. (2020) gave credence to this statement by showing, through empirical measurements, how sharpness can predict generalization. Dinh et al. (2017) questioned this conjecture by showing that most notions of flatness are problematic for deep models and can not be tied to generalization. Their argument relied on the observation that one can reparametrize a model and increase arbitrarily the sharpness of its minima, without changing the function it implements. However, the measures of flatness considered by Dinh et al. (2017) were the trace (sum of eigenvalues) or spectral norm (maximal eigenvalue) of the Hessian; they never considered the separation of the outliers from the bulk in the spectrum of the Hessian which, in light of our paper, should correlate with generalization.

10. Conclusions

There exist many fundamental objects associated with deep neural networks. No one has any intuition about their properties, structure or behaviour. Researchers therefore started extracting from them descriptive statistics like eigenvalues. It is remarkable such measurements are even possible, given how high-dimensional some of these objects are. After measuring the eigenvalues, one starts seeing a great deal of structure, which is curiously

explicit, consisting of C outliers, C^2 secondary outliers, and a main-bulk. Researchers pointed to the existence of some of the structure or its traces. However, it has been an open question as to what is causing this structure to appear and how to exploit it. This paper shows there exists a specific highly organized structure and those initial observations can be expected to persist everywhere. These are not artifacts but deeply significant observations and understanding them is important for understanding deep learning. This is not a spandrel, it is a clue of deep significance.

Acknowledgments

We are grateful to David L. Donoho for helpful discussions and for providing comments on this manuscript. We also thank the anonymous referees for their helpful comments. This work was partially supported by NSF DMS 1407813, 1418362, and 1811614 and by private donors. Some of the computing for this project was performed on the Sherlock cluster at Stanford University; we thank the Stanford Research Computing Center for providing computational resources and support that enabled our research. Some of this project was also performed on Google Cloud Platform: thanks to Google Cloud Platform Education Grants Program for research credits that supplemented this work. Moreover, we thank Riccardo Murri and Hatef Monajemi for their extensive help with the Elasticluster and ClusterJob frameworks, respectively.

Appendix A. Cross-class structure in G

Recall the definition of $\boldsymbol{G}$,

$$\boldsymbol{G} = \underset{i,c}{\text{Ave}} \left\{ \frac{\partial f(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}}^\top \frac{\partial^2 \ell(\boldsymbol{x}_{i,c}, \boldsymbol{y}_{i,c})}{\partial \boldsymbol{f}^2} \frac{\partial f(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}} \right\}.$$

Plugging the Hessian of multinomial logistic regression Böhning (1992), we obtain

$$\boldsymbol{G} = \underset{i,c}{\text{Ave}} \left\{ \frac{\partial f(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}}^\top \left(\text{diag}(\boldsymbol{p}_{i,c}) - \boldsymbol{p}_{i,c}\boldsymbol{p}_{i,c}^\top \right) \frac{\partial f(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}} \right\}.$$

The above can be shown to be equivalent to

$$\boldsymbol{G} = \underset{i,c}{\text{Ave}} \left\{ \sum_{c'} p_{i,c,c'} \left(\frac{\partial f_{c'}(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}} - \sum_{c'} p_{i,c,c'} \frac{\partial f_{c'}(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}} \right)^\top \left(\frac{\partial f_{c'}(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}} - \sum_{c'} p_{i,c,c'} \frac{\partial f_{c'}(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}} \right) \right\}.$$

Define the p-dimensional vector,

$$\boldsymbol{g}_{i,c,c'} = \frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \boldsymbol{\theta}},$$

and note that

$$\begin{aligned}
\boldsymbol{g}_{i,c,c'} &= \frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \boldsymbol{\theta}} \\
&= \frac{\partial f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}^\top \frac{\partial \ell(f(\boldsymbol{x}_{i,c}; \boldsymbol{\theta}), \boldsymbol{y}_{c'})}{\partial \boldsymbol{f}} \\
&= \frac{\partial f(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}}^\top (\boldsymbol{y}_{c'} - \boldsymbol{p}_{i,c}) \\
&= \frac{\partial f_{c'}(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}} - \sum_{c'} p_{i,c,c'} \frac{\partial f_{c'}(\boldsymbol{x}_{i,c})}{\partial \boldsymbol{\theta}}.
\end{aligned}$$

Plugging the above expression into the definition of $\boldsymbol{G}$, we get

$$\boldsymbol{G} = \underset{i,c}{\text{Ave}} \left\{ \sum_{c'} p_{i,c,c'} \boldsymbol{g}_{i,c,c'} \boldsymbol{g}_{i,c,c'}^\top \right\},$$

or equally

$$\boldsymbol{G} = \sum_{i,c,c'} w_{i,c,c'} \boldsymbol{g}_{i,c,c'} \boldsymbol{g}_{i,c,c'}^\top.$$

A.1 First decomposition

Recall the following definitions:

$$\begin{aligned}
\boldsymbol{g}_{c,c'} &= \sum_i \pi_{i,c,c'} \boldsymbol{g}_{i,c,c'} \\
\boldsymbol{G}_{\text{within},c,c'} &= \sum_i \pi_{i,c,c'} (\boldsymbol{g}_{i,c,c'} - \boldsymbol{g}_{c,c'})(\boldsymbol{g}_{i,c,c'} - \boldsymbol{g}_{c,c'})^\top \\
\pi_{i,c,c'} &= \frac{w_{i,c,c'}}{w_{c,c'}} \\
w_{c,c'} &= \sum_i w_{i,c,c'}.
\end{aligned}$$

Note that

$$\begin{aligned}
\boldsymbol{G} &= \sum_{i,c,c'} w_{i,c,c'} \boldsymbol{g}_{i,c,c'} \boldsymbol{g}_{i,c,c'}^\top \\
&= \sum_{c,c'} w_{c,c'} \frac{1}{w_{c,c'}} \sum_i w_{i,c,c'} \boldsymbol{g}_{i,c,c'} \boldsymbol{g}_{i,c,c'}^\top \\
&= \sum_{c,c'} w_{c,c'} \sum_i \pi_{i,c,c'} \boldsymbol{g}_{i,c,c'} \boldsymbol{g}_{i,c,c'}^\top \\
&= \sum_{c,c'} w_{c,c'} (\boldsymbol{g}_{c,c'} \boldsymbol{g}_{c,c'}^\top + \boldsymbol{G}_{\text{within},c,c'}) \\
&= \sum_{c,c'} w_{c,c'} \boldsymbol{g}_{c,c'} \boldsymbol{g}_{c,c'}^\top + \sum_{c,c'} w_{c,c'} \boldsymbol{G}_{\text{within},c,c'} \\
&= \sum_{\substack{c,c' \\ c \neq c'}} w_{c,c'} \boldsymbol{g}_{c,c'} \boldsymbol{g}_{c,c'}^\top + \sum_c w_{c,c} \boldsymbol{g}_{c,c} \boldsymbol{g}_{c,c}^\top + \sum_{c,c'} w_{c,c'} \boldsymbol{G}_{\text{within},c,c'}.
\end{aligned}$$

In what follows, we further decompose only the first summation in the above equation.

A.2 Second decomposition

Recall the following definitions:

$$\begin{aligned}
\boldsymbol{g}_c &= \sum_{c' \neq c} \pi_{c,c'} \boldsymbol{g}_{c,c'} \\
\boldsymbol{G}_{\text{cross},c} &= \sum_{c' \neq c} \pi_{c,c'} (\boldsymbol{g}_{c,c'} - \boldsymbol{g}_{c'})(\boldsymbol{g}_{c,c'} - \boldsymbol{g}_{c'})^\top \\
\pi_{c,c'} &= \frac{w_{c,c'}}{w_c} \\
w_c &= \sum_{c' \neq c} w_{c,c'}.
\end{aligned}$$

We have

$$\begin{aligned}\sum_{\substack{c,c'\\c\neq c'}} w_{c,c'}\boldsymbol{g}_{c,c'}\boldsymbol{g}_{c,c'}^\top &= \sum_c \sum_{c'\neq c} w_{c,c'}\boldsymbol{g}_{c,c'}\boldsymbol{g}_{c,c'}^\top \\ &= \sum_c w_c \frac{1}{w_c}\sum_{c'\neq c} w_{c,c'}\boldsymbol{g}_{c,c'}\boldsymbol{g}_{c,c'}^\top \\ &= \sum_c w_c \sum_{c'\neq c} \pi_{c,c'}\boldsymbol{g}_{c,c'}\boldsymbol{g}_{c,c'}^\top \\ &= \sum_c w_c(\boldsymbol{g}_c\boldsymbol{g}_c^\top + \boldsymbol{G}_{\text{cross},c}) \\ &= \sum_c w_c\boldsymbol{g}_c\boldsymbol{g}_c^\top + \sum_c w_c\boldsymbol{G}_{\text{cross},c}.\end{aligned}$$

A.3 Combination

Combining all the expressions from the previous subsections, we get

$$\boldsymbol{G} = \underbrace{\sum_c w_c\boldsymbol{g}_c\boldsymbol{g}_c^\top}_{\boldsymbol{G}_{\text{class}}} + \underbrace{\sum_c w_c\boldsymbol{G}_{\text{cross},c}}_{\boldsymbol{G}_{\text{cross}}} + \underbrace{\sum_{c,c'} w_{c,c'}\boldsymbol{G}_{\text{within},c,c'}}_{\boldsymbol{G}_{\text{within}}} + \underbrace{\sum_c w_{c,c}\boldsymbol{g}_{c,c}\boldsymbol{g}_{c,c}^\top}_{\boldsymbol{G}_{c=c'}}. \tag{A.1}$$

Appendix B. Multinomial logistic regression

Lemma 8.1 (Expected FIM of multinomial logistic regression trained on $\mathcal{CCM}$). *The expected FIM of multinomial logistic regression, with symmetric probabilities, trained on $\mathcal{CCM}(D,C)$, is given by:*

$$\mathbb{E}\,\boldsymbol{G} = \frac{s}{C}\cdot \text{blkdiag}(\boldsymbol{U}_1,\ldots,\boldsymbol{U}_C,\boldsymbol{0}_{DC-C^2}) + \boldsymbol{I}\otimes\bar{\boldsymbol{U}},$$

where

$$\begin{aligned}s &= t^2\\ \boldsymbol{U}_c &= \text{diag}(\boldsymbol{p}_{\cdot,c}) - \boldsymbol{p}_{\cdot,c}\boldsymbol{p}_{\cdot,c}^\top,\\ \bar{\boldsymbol{U}} &= \operatorname*{Ave}_c \boldsymbol{U}_c,\end{aligned}$$

and the operator blkdiag$(\cdot)$ *forms a block diagonal matrix with $\boldsymbol{U}_c$ in its c'th block and zeros elsewhere.*

Proof Recall the definition of $\boldsymbol{G}$:

$$\boldsymbol{G} = \operatorname*{Ave}_{i,c}\left\{(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C)(\text{diag}(\boldsymbol{p}_{i,c}) - \boldsymbol{p}_{i,c}\boldsymbol{p}_{i,c}^\top)(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C)^\top\right\}.$$

Using our symmetric probabilities assumption, there exist $C\times C$ matrices $\{\boldsymbol{U}_c\}_c$ for which

$$\boldsymbol{G} = \operatorname*{Ave}_{i,c}\left\{(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C)\boldsymbol{U}_c(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C)^\top\right\}.$$

Using the property of the Kronecker product, $(\boldsymbol{A}\otimes\boldsymbol{B})(\boldsymbol{C}\otimes\boldsymbol{D})=(\boldsymbol{AC})\otimes(\boldsymbol{BD})$, the expression in the above average can be simplified into:

$$\begin{aligned}&\left(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C\right)\boldsymbol{U}_c\left(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C\right)^\top\\=&\left(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C\right)\left(1\otimes\boldsymbol{U}_c\right)\left(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C\right)^\top\\=&\left(\boldsymbol{x}_{i,c}\otimes\boldsymbol{U}_c\right)\left(\boldsymbol{x}_{i,c}\otimes\boldsymbol{I}_C\right)^\top\\=&(\boldsymbol{x}_{i,c}\boldsymbol{x}_{i,c}^\top)\otimes\boldsymbol{U}_c.\end{aligned}$$

Plugging this expression into the previous equation, we get

$$\boldsymbol{G}=\underset{i,c}{\text{Ave}}\left\{(\boldsymbol{x}_{i,c}\boldsymbol{x}_{i,c}^\top)\otimes\boldsymbol{U}_c\right\}.$$

Plugging our assumption that $\boldsymbol{x}_{i,c}=t\boldsymbol{e}_c+\boldsymbol{z}_{i,c}$, we obtain

$$\begin{aligned}\boldsymbol{G}=&t^2\underset{i,c}{\text{Ave}}\left\{(\boldsymbol{e}_c\boldsymbol{e}_c^\top)\otimes\boldsymbol{U}_c\right\}\\&+t\underset{i,c}{\text{Ave}}\left\{(\boldsymbol{e}_c\boldsymbol{z}_{i,c}^\top)\otimes\boldsymbol{U}_c\right\}\\&+t\underset{i,c}{\text{Ave}}\left\{(\boldsymbol{z}_{i,c}\boldsymbol{e}_c^\top)\otimes\boldsymbol{U}_c\right\}\\&+\underset{i,c}{\text{Ave}}\left\{(\boldsymbol{z}_{i,c}\boldsymbol{z}_{i,c}^\top)\otimes\boldsymbol{U}_c\right\}.\end{aligned}$$

Taking the expectation of both sides over $\boldsymbol{z}_{i,c}$, we get

$$\mathbb{E}\,\boldsymbol{G}=t^2\underset{c}{\text{Ave}}\left\{(\boldsymbol{e}_c\boldsymbol{e}_c^\top)\otimes\boldsymbol{U}_c\right\}+\underset{c}{\text{Ave}}\left\{\boldsymbol{I}\otimes\boldsymbol{U}_c\right\}.$$

The above can be written concisely as follows:

$$\mathbb{E}\,\boldsymbol{G}=\frac{s}{C}\,\text{blkdiag}(\boldsymbol{U}_1,\ldots,\boldsymbol{U}_C,\boldsymbol{0}_{DC-C^2})+\boldsymbol{I}\otimes\underset{c}{\text{Ave}}\,\boldsymbol{U}_c.$$

■

Lemma B.1 *Consider the $C\times C$ matrix,*

$$\boldsymbol{A}=\begin{bmatrix}a&b&\ldots&b\\b&a&\ldots&b\\\vdots&\vdots&\ddots&\vdots\\b&b&\ldots&a\end{bmatrix}.$$

Its eigenvalues are given by:

$$\lambda(\boldsymbol{A})=\begin{Bmatrix}a+b(C-1), & \text{for } i=1\\a-b, & \text{for } 1<i\leq C\end{Bmatrix}.$$

Proof Notice that

$$\boldsymbol{A}\frac{1}{\sqrt{C}}\mathbb{1} = (a + b(C-1))\frac{1}{\sqrt{C}}\mathbb{1}.$$

As such, $\frac{1}{\sqrt{C}}\mathbb{1}$ is an eigenvector with an eigenvalue $a + b(C-1)$. Recall that the trace of a matrix is equal to the sum of its eigenvalues. Notice that the rest of the eigenvalues are all the same. As such, their value is equal to

$$\frac{\text{Tr}\{\boldsymbol{A}\} - (a + b(C-1))}{C-1} = \frac{aC - (a + b(C-1))}{C-1} = a - b.$$

■

Lemma B.2 *Consider the $C \times C$ matrix,*

$$\boldsymbol{B} = \left[\begin{array}{c|cccc} a & b & b & \dots & b \\ \hline b & d & c & \dots & c \\ b & c & d & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ b & c & \dots & \dots & d \end{array}\right].$$

Its eigenvalues are given by:

$$\lambda(\boldsymbol{B}) = \left\{\begin{array}{ll} (T+\Delta)/2, & \textit{for } i = 1 \\ (T-\Delta)/2, & \textit{for } i = 2 \\ d - e, & \textit{for } 2 < i \leq C \end{array}\right\}.$$

where

$$\begin{aligned} T =& a + d + e(C-2) \\ D =& a\left(d + e\left(C-2\right)\right) - b^2(C-1) \\ \Delta =& \sqrt{T^2 - 4D}. \end{aligned}$$

Proof Consider the $(C-1) \times (C-1)$ matrix,

$$\boldsymbol{A} = \begin{bmatrix} d & c & \dots & c \\ c & d & \dots & c \\ \vdots & \vdots & \ddots & \vdots \\ c & c & \dots & d \end{bmatrix}.$$

Its eigenvalues are given by:

$$\lambda(\boldsymbol{A}) = \left\{\begin{array}{ll} d + c(C-1), & \text{for } i = 1 \\ d - c, & \text{for } 1 < i \leq C-1 \end{array}\right\}.$$

Denote by $\boldsymbol{v}_3, \dots, \boldsymbol{v}_C$ the $C-2$ eigenvectors corresponding to the eigenvalue $d - c$. Define the orthonormal basis:

$$\boldsymbol{V} = \left[\boldsymbol{e}_1, \frac{1}{\sqrt{C}}\tilde{\mathbb{1}}, \boldsymbol{v}_3, \dots, \boldsymbol{v}_C\right],$$

where $\tilde{\mathbb{1}}$ is a C-dimensional vector of ones, except in the first coordinate, where it is equal to zero. In other words,

$$\tilde{\mathbb{1}} = [0, 1, \ldots, 1]\,.$$

Notice that

$$\boldsymbol{V}^T\boldsymbol{B}\boldsymbol{V} = \left[\begin{array}{cc|cccc} a & b\sqrt{C-1} & 0 & 0 & \ldots & 0 \\ b\sqrt{C-1} & d+c(C-2) & 0 & 0 & \ldots & 0 \\ \hline 0 & 0 & d-c & 0 & \ldots & 0 \\ 0 & 0 & 0 & d-c & & 0 \\ \vdots & & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & d-c \end{array}\right].$$

The diagonal values in the above matrix are obtained through the following equations

$$\begin{aligned} \boldsymbol{e}_1^\top \boldsymbol{B}\boldsymbol{e}_1 =& a \\ \frac{1}{\sqrt{C}}\tilde{\mathbb{1}}^\top \boldsymbol{B}\frac{1}{\sqrt{C}}\tilde{\mathbb{1}}_1 =& d + c(C-2) \\ \boldsymbol{v}_i\boldsymbol{B}\boldsymbol{v}_i =& d-c, \quad i \geq 3. \end{aligned}$$

The off-diagonal values are obtained by noticing that

$$\begin{aligned} \boldsymbol{e}_1^\top \boldsymbol{B}\frac{1}{\sqrt{C}}\tilde{\mathbb{1}} =& b\sqrt{C-1} \\ \boldsymbol{e}_1^\top \boldsymbol{B}\boldsymbol{v}_i =& 0, \quad i \geq 3 \\ \boldsymbol{v}_i^\top \boldsymbol{B}\boldsymbol{v}_j =& 0, \quad i,j \geq 3. \end{aligned}$$

The above implies that $d-c$ is an eigenvalue with multiplicity $C-2$. The other two eigenvalues are equal to the eigenvalues of the matrix

$$\boldsymbol{C} = \left[\begin{array}{cc} a & b\sqrt{C-1} \\ b\sqrt{C-1} & d+e(C-2) \end{array}\right].$$

Recall that the trace of the matrix is equal to the sum of its eigenvalues and the determinant to the multiplication of the eigenvalues. As such,

$$\begin{aligned} a + d + e(C-2) &= \mathrm{Tr}\{\boldsymbol{C}\} = \lambda_1 + \lambda_2 \\ a(d+e(C-2)) - b^2(C-1) &= |\boldsymbol{C}| = \lambda_1\lambda_2. \end{aligned}$$

Notice that

$$\lambda_2 = \frac{|\boldsymbol{C}|}{\lambda_1}.$$

Plugging the above into the equation of the trace, we get

$$\mathrm{Tr}\{\boldsymbol{C}\} = \lambda_1 + \frac{|\boldsymbol{C}|}{\lambda_1}.$$

Multiplying both sides by λ_1 and rearranging the terms, we obtain

$$0 = \lambda_1^2 - \mathrm{Tr}\{\boldsymbol{C}\}\lambda_1 + |\boldsymbol{C}|,$$

the solution of which is

$$\lambda_1 = \frac{\mathrm{Tr}\{C\} \pm \sqrt{\mathrm{Tr}\{\boldsymbol{C}\}^2 - 4|\boldsymbol{C}|}}{2}.$$

Defining

$$\Delta = \sqrt{\mathrm{Tr}\{\boldsymbol{C}\}^2 - 4|\boldsymbol{C}|},$$

we obtain

$$\lambda_1 = \frac{\mathrm{Tr}\{C\} \pm \Delta}{2}.$$

■

Theorem 8.1 (Spectrum of expected FIM of multinomial logistic regression trained on $\mathcal{CCM}$). *The spectrum of the expected FIM of multinomial logistic regression, with symmetric probabilities, trained on $\mathcal{CCM}(D, C)$, is given by:*

$$\lambda_i\left(\mathbb{E}\,\boldsymbol{G}\right) = \begin{cases} \frac{\alpha}{C-1}\left(s(1-\alpha) + \left(2 - \alpha\frac{C}{C-1}\right)\right), & \text{for } 1 \le i \le C \\ \frac{\alpha}{C-1}\left(\frac{s}{C} + \left(2 - \alpha\frac{C}{C-1}\right)\right), & \text{for } C < i \le C(C-1) \\ \frac{\alpha}{C-1}\left(2 - \alpha\frac{C}{C-1}\right), & \text{for } C(C-1) < i \le D(C-1) \\ 0, & \text{for } D(C-1) < i \le DC \end{cases}.$$

Proof Consider without loss of generality the matrix $\boldsymbol{U}_1$. Using the symmetric probabilities assumption, the (c, c') entry of this matrix is given by:

$$\boldsymbol{U}_1(c', c'') = \underset{i}{\mathrm{Ave}}\left\{\delta_{\{c'=c''\}}p_{i,1,c'} - p_{i,1,c'}p_{i,1,c''}\right\}.$$

As such, the whole matrix equals

$$\boldsymbol{U}_1 = \left[\begin{array}{c|cccc} 1-\alpha & 0 & 0 & \dots & 0 \\ \hline 0 & \frac{\alpha}{C-1} & 0 & \dots & 0 \\ 0 & 0 & \frac{\alpha}{C-1} & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \dots & \frac{\alpha}{C-1} \end{array}\right] - \left[\begin{array}{c|cccc} (1-\alpha)^2 & \frac{\alpha(1-\alpha)}{C-1} & \frac{\alpha(1-\alpha)}{C-1} & \dots & \frac{\alpha(1-\alpha)}{C-1} \\ \hline \frac{\alpha(1-\alpha)}{C-1} & (\frac{\alpha}{C-1})^2 & (\frac{\alpha}{C-1})^2 & \dots & (\frac{\alpha}{C-1})^2 \\ \frac{\alpha(1-\alpha)}{C-1} & (\frac{\alpha}{C-1})^2 & (\frac{\alpha}{C-1})^2 & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ \frac{\alpha(1-\alpha)}{C-1} & (\frac{\alpha}{C-1})^2 & \dots & \dots & (\frac{\alpha}{C-1})^2 \end{array}\right].$$

Simplifying the expressions, we get

$$\boldsymbol{U}_1 = \left[\begin{array}{c|ccc} \alpha(1-\alpha) & -\frac{\alpha(1-\alpha)}{C-1} & -\frac{\alpha(1-\alpha)}{C-1} & \cdots & -\frac{\alpha(1-\alpha)}{C-1} \\ \hline -\frac{\alpha(1-\alpha)}{C-1} & \frac{\alpha}{C-1}\left(1-\frac{\alpha}{C-1}\right) & -(\frac{\alpha}{C-1})^2 & \cdots & -(\frac{\alpha}{C-1})^2 \\ -\frac{\alpha(1-\alpha)}{C-1} & -(\frac{\alpha}{C-1})^2 & \frac{\alpha}{C-1}\left(1-\frac{\alpha}{C-1}\right) & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ -\frac{\alpha(1-\alpha)}{C-1} & -(\frac{\alpha}{C-1})^2 & \cdots & \cdots & \frac{\alpha}{C-1}\left(1-\frac{\alpha}{C-1}\right) \end{array}\right].$$

Notice that

$$\bar{\boldsymbol{U}} = \begin{bmatrix} a & b & \dots & b \\ b & a & \dots & b \\ \vdots & \vdots & \ddots & \vdots \\ b & b & \dots & a \end{bmatrix},$$

where

$$\begin{aligned} a =& \frac{1}{C}\alpha(1-\alpha) + \frac{C-1}{C}\frac{\alpha}{C-1}\left(1-\frac{\alpha}{C-1}\right) \\ =& \frac{\alpha}{C}\left(2-\alpha\frac{C}{C-1}\right), \end{aligned}$$

and

$$\begin{aligned} b =& \frac{2}{C}\frac{-\alpha(1-\alpha)}{C-1} + \frac{C-2}{C}\frac{-\alpha^2}{(C-1)^2} \\ =& -\frac{\alpha}{C(C-1)}\left(2-\alpha\frac{C}{C-1}\right). \end{aligned}$$

Using the previous lemma, the eigenvalues of $\bar{\boldsymbol{U}}$ are given by

$$\lambda(\bar{\boldsymbol{U}}) = \left\{ \begin{array}{ll} a+b(C-1), & \text{for } i=1 \\ a-b, & \text{for } 1<i\le C \end{array} \right\},$$

or equally,

$$\lambda(\bar{\boldsymbol{U}}) = \left\{ \begin{array}{ll} 0, & \text{for } i=1 \\ \frac{\alpha}{C-1}\left(2-\alpha\frac{C}{C-1}\right), & \text{for } 1<i\le C \end{array} \right\}.$$

Recalling that

$$\mathbb{E}\,\boldsymbol{G} = \frac{s}{C}\,\mathrm{blkdiag}(\boldsymbol{U}_1,\dots,\boldsymbol{U}_C,\boldsymbol{0}_{(D-C)C}) + \boldsymbol{I}\otimes\bar{\boldsymbol{U}},$$

we get that $\mathbb{E}\,\boldsymbol{G}$ has $(D-C)(C-1)$ eigenvalues equal to

$$\frac{\alpha}{C-1}\left(2-\alpha\frac{C}{C-1}\right), \tag{B.1}$$

and also $D - C$ eigenvalues equal to zero. Next, notice that

$$\frac{s}{C}\boldsymbol{U}_1 + \bar{\boldsymbol{U}} = \left[\begin{array}{c|cccc} a & b & b & \dots & b \\ \hline b & d & e & \dots & e \\ b & e & d & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ b & e & \dots & \dots & d \end{array}\right],$$

where

$$\begin{aligned} a =& \frac{s}{C}\alpha(1-\alpha) + \frac{\alpha}{C}\left(2 - \alpha\frac{C}{C-1}\right) \\ b =& -\frac{s}{C}\frac{\alpha(1-\alpha)}{C-1} - \frac{\alpha}{C(C-1)}\left(2 - \alpha\frac{C}{C-1}\right) \\ e =& -\frac{s}{C}\left(\frac{\alpha}{C-1}\right)^2 - \frac{\alpha}{C(C-1)}\left(2 - \alpha\frac{C}{C-1}\right) \\ d =& \frac{s}{C}\frac{\alpha(C-1-\alpha)}{(C-1)^2} + \frac{\alpha}{C}\left(2 - \alpha\frac{C}{C-1}\right), \end{aligned}$$

or equally,

$$\begin{aligned} a =& \frac{\alpha}{C}\left(s(1-\alpha) + \left(2 - \alpha\frac{C}{C-1}\right)\right) \\ b =& -\frac{\alpha}{C(C-1)}\left(s(1-\alpha) + \left(2 - \alpha\frac{C}{C-1}\right)\right) \\ e =& -\frac{\alpha}{C(C-1)}\left(s\frac{\alpha}{C-1} + \left(2 - \alpha\frac{C}{C-1}\right)\right) \\ d =& \frac{\alpha}{C}\left(s\frac{C-1-\alpha}{(C-1)^2} + \left(2 - \alpha\frac{C}{C-1}\right)\right). \end{aligned}$$

Using the previous lemma, its eigenvalues are given by:

$$\lambda\left(\frac{s}{C}\boldsymbol{U}_1 + \bar{\boldsymbol{U}}\right) = \left\{\begin{array}{ll} (T+\Delta)/2, & \text{for } i = 1 \\ (T-\Delta)/2, & \text{for } i = 2 \\ d - e, & \text{for } 2 < i \leq C \end{array}\right\}. \tag{B.2}$$

where

$$\begin{aligned} T =& a + d + e(C-2) \\ D =& a\left(d + e\left(C-2\right)\right) - b^2(C-1) \\ \Delta =& \sqrt{T^2 - 4D}. \end{aligned}$$

Plugging the values of d and e and simplifying the expressions, we get

$$\begin{aligned} d-e =& \frac{\alpha}{C}\left(s\frac{C-1-\alpha}{(C-1)^2}+\left(2-\alpha\frac{C}{C-1}\right)\right) && \text{(B.3)}\\ &+\frac{\alpha}{C(C-1)}\left(s\frac{\alpha}{C-1}+\left(2-\alpha\frac{C}{C-1}\right)\right) && \text{(B.4)}\\ =& \frac{\alpha}{C-1}\left(\frac{s}{C}+\left(2-\alpha\frac{C}{C-1}\right)\right). && \text{(B.5)} \end{aligned}$$

Plugging the values into the trace, we obtain

$$\begin{aligned} T =& \frac{\alpha}{C}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right)\\ &+\frac{\alpha}{C}\left(s\frac{C-1-\alpha}{(C-1)^2}+\left(2-\alpha\frac{C}{C-1}\right)\right)\\ &-\frac{\alpha}{C(C-1)}\left(s\frac{\alpha}{C-1}+\left(2-\alpha\frac{C}{C-1}\right)\right)(C-2)\\ =& \frac{s\alpha(1-\alpha)}{C-1}+\frac{\alpha}{C-1}\left(2-\alpha\frac{C}{C-1}\right)\\ =& \frac{\alpha}{C-1}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right). \end{aligned}$$

Notice that

$$\begin{aligned} & d+e\,(C-2)\\ =& \frac{\alpha}{C}\left(s\frac{C-1-\alpha}{(C-1)^2}+\left(2-\alpha\frac{C}{C-1}\right)\right)\\ &-\frac{\alpha}{C(C-1)}\left(s\frac{\alpha}{C-1}+\left(2-\alpha\frac{C}{C-1}\right)\right)(C-2)\\ =& \frac{\alpha}{C(C-1)}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right). \end{aligned}$$

As such,

$$\begin{aligned} & a\,(d+e\,(C-2))\\ =& \frac{\alpha}{C}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right)\\ &\times\frac{\alpha}{C(C-1)}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right)\\ =& \frac{\alpha^2}{C^2(C-1)}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right)^2. \end{aligned}$$

Notice also that

$$\begin{aligned}
&b^2(C-1)\\
=&\left(\frac{\alpha}{C(C-1)}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right)\right)^2(C-1)\\
=&\frac{\alpha^2}{C^2(C-1)}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right)^2.
\end{aligned}$$

Combining the two previous expressions, we get the determinant is equal to zero,

$$\begin{aligned}
D=&a\left(d+e\left(C-2\right)\right)-b^2(C-1)\\
=&\frac{\alpha^2}{C^2(C-1)}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right)^2\\
&-\frac{\alpha^2}{C^2(C-1)}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right)^2\\
=&0.
\end{aligned}$$

Hence,

$$\Delta=\sqrt{T^2-4D}=T.$$

One of the eigenvalues of $\frac{s}{C}\boldsymbol{U}_1+\bar{\boldsymbol{U}}$ is zero, since

$$\frac{T-\Delta}{2}=\frac{T-T}{2}=0, \tag{B.6}$$

while the other is equal to

$$\frac{T+\Delta}{2}=\frac{T+T}{2}=T=\frac{\alpha}{C-1}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right). \tag{B.7}$$

Plugging Equations (B.3), (B.7) and (B.6) into Equation (B.2), we get

$$\lambda\left(\frac{s}{C}\boldsymbol{U}_1+\bar{\boldsymbol{U}}\right)=\left\{\begin{array}{ll}\frac{\alpha}{C-1}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right), & \text{for } i=1\\ \frac{\alpha}{C-1}\left(\frac{s}{C}+\left(2-\alpha\frac{C}{C-1}\right)\right), & \text{for } 1<i<C\\ 0, & \text{for } i=C\end{array}\right\}.$$

Recall that the spectrum of $\frac{s}{C}\boldsymbol{U}_c+\bar{\boldsymbol{U}}$ does not depend on c. As such, the spectrum of $\frac{s}{C}\,\mathrm{blkdiag}(\boldsymbol{U}_1,\ldots,\boldsymbol{U}_C)+\boldsymbol{I}\otimes\bar{\boldsymbol{U}}$ is obtained by simply multiplying the multiplicity of each eigenvalue in $\frac{s}{C}\boldsymbol{U}_1+\bar{\boldsymbol{U}}$ by C, i.e.,

$$\begin{aligned}
&\lambda\left(\frac{s}{C}\,\mathrm{blkdiag}(\boldsymbol{U}_1,\ldots,\boldsymbol{U}_C)+\boldsymbol{I}\otimes\bar{\boldsymbol{U}}\right)\\
=&\left\{\begin{array}{ll}\frac{\alpha}{C-1}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right), & \text{for } 1\le i\le C\\ \frac{\alpha}{C-1}\left(\frac{s}{C}+\left(2-\alpha\frac{C}{C-1}\right)\right), & \text{for } C<i\le C(C-1)\\ 0, & \text{for } C(C-1)<i\le C^2\end{array}\right\}.
\end{aligned}$$

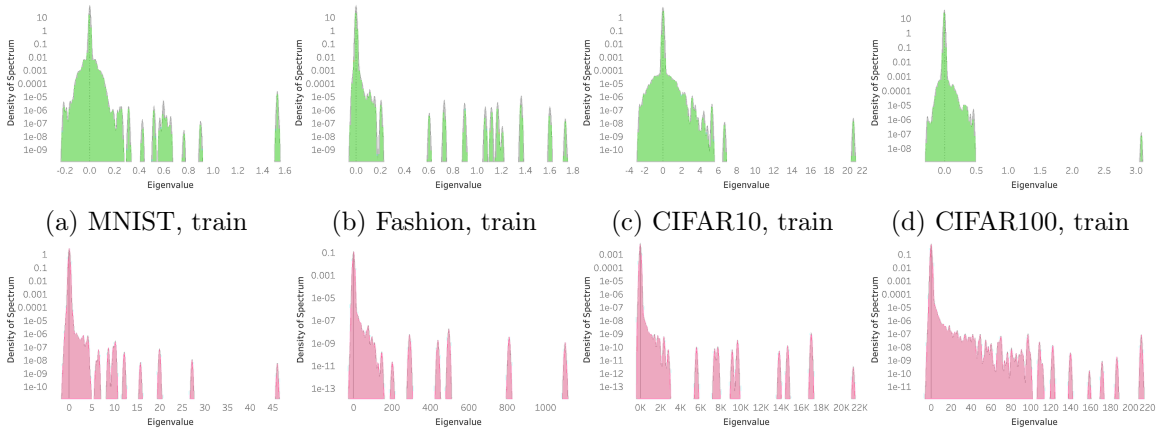
Combining the above with Equation (B.1), we conclude

$$\lambda_i\left(\mathbb{E}\,\boldsymbol{G}\right) = \lambda_i\left(\frac{s}{C}\,\mathrm{blkdiag}(\boldsymbol{U}_1,\ldots,\boldsymbol{U}_C,\boldsymbol{0}_{(D-C)C}) + \boldsymbol{I}\otimes\bar{\boldsymbol{U}}\right)$$
$$= \left\{\begin{array}{ll} \frac{\alpha}{C-1}\left(s(1-\alpha)+\left(2-\alpha\frac{C}{C-1}\right)\right), & \text{for } 1\le i\le C \\ \frac{\alpha}{C-1}\left(\frac{s}{C}+\left(2-\alpha\frac{C}{C-1}\right)\right), & \text{for } C<i\le C(C-1) \\ \frac{\alpha}{C-1}\left(2-\alpha\frac{C}{C-1}\right), & \text{for } C(C-1)<i\le D(C-1) \\ 0, & \text{for } D(C-1)<i\le DC \end{array}\right\}.$$

■

Appendix C. Tools from numerical linear algebra

Our approach for approximating the spectrum of the Hessian (and similarly other quantities) builds on the survey of Lin et al. (2016), which discussed several different methods for approximating the density of the spectrum of large linear operators; many of which were first developed by physicists and chemists in quantum mechanics starting from the 1970's (Ducastelle and Cyrot-Lackmann, 1970; Wheeler and Blumstein, 1972; Turek, 1988; Drabold and Sankey, 1993). From the methods presented therein, we implemented and tested two: the Lanczos method and KPM. From our experience Lanczos was effective and useful, while KPM was temperamental and problematic. As such, we focus in this work on Lanczos.



(a) MNIST, train (b) Fashion, train (c) CIFAR10, train (d) CIFAR100, train

(e) MNIST, test (f) Fashion, test (g) CIFAR10, test (h) CIFAR100, test

Figure 14: **Spectrum of the Hessian for VGG11 trained on various datasets.** Each panel corresponds to a different famous dataset in deep learning. The spectrum was approximated using LanczosApproxSpec. Unlike the figure in the main manuscript, the top-C eigenspace was not removed using LowRankDeflation.

C.1 SlowLanczos

The Lanczos algorithm (Lanczos, 1950) computes the spectrum of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ by first reducing it to a tridiagonal form $\boldsymbol{T}_p \in \mathbb{R}^{p\times p}$ and then computing the spectrum of that matrix instead. The motivation is that computing the spectrum of a tridiagonal matrix is very efficient, requiring only $O(p^2)$ operations. The algorithm works by progressively building an adapted orthonormal basis $\boldsymbol{V}_m \in \mathbb{R}^{p\times m}$ that satisfies at each iteration the relation $\boldsymbol{V}_m^\top \boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{T}_m$, where $\boldsymbol{T}_m \in \mathbb{R}^{m\times m}$ is a tridiagonal matrix. For completeness, we summarize its main steps in Algorithm 1.

Algorithm 1: SlowLanczos($\boldsymbol{A}$)

Input: Linear operator $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ with spectrum in the range $[-1, 1]$.
Result: Eigenvalues and eigenvectors of the tridiagonal matrix $\boldsymbol{T}_p$.
for $m = 1, \ldots, p$ **do**
  if $m == 1$ **then**
    sample $\boldsymbol{v} \sim \mathcal{N}(0, \boldsymbol{I})$;
    $\boldsymbol{v}_1 = \frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|_2}$;
    $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{v}_1$;
  else
    $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{v}_m - \beta_{m-1}\boldsymbol{v}_{m-1}$;
  end
  $\alpha_m = \boldsymbol{v}_m^\top \boldsymbol{w}$;
  $\boldsymbol{w} = \boldsymbol{w} - \alpha_m \boldsymbol{v}_m$;
  `/* reorthogonalization */`
  $\boldsymbol{w} = \boldsymbol{w} - \boldsymbol{V}_m \boldsymbol{V}_m^\top \boldsymbol{w}$;
  $\beta_m = \|\boldsymbol{w}\|_2$;
  $\boldsymbol{v}_{m+1} = \frac{\boldsymbol{w}}{\beta_m}$;
end

$$\boldsymbol{T}_p = \begin{bmatrix} \alpha_1, & \beta_1, & & & \\ \beta_1, & \alpha_2, & \beta_2, & & \\ & \beta_2, & \alpha_3, & & \\ & & & \ddots & \beta_{p-1} \\ & & & \beta_{p-1} & \alpha_p \end{bmatrix};$$

$\{\theta_m\}_{m=1}^p, \{\boldsymbol{y}_m\}_{m=1}^p = \mathrm{eig}(\boldsymbol{T}_p)$;
return $\{\theta_m\}_{m=1}^p, \{\boldsymbol{y}_m\}_{m=1}^p$;

C.2 Complexity of SlowLanczos

Assume without loss of generality we are computing the spectrum of the train Hessian. Each of the p iterations of the algorithm requires a single Hessian-vector multiplication, incurring $O(Np)$ complexity. The complexity due to all Hessian-vector multiplications is therefore $O(Np^2)$. The m'th iteration also requires a reorthogonalization step, which computes the inner product of m vectors of length p and costs $O(mp)$ complexity. Summing this over

the iterations, $m = 1, \ldots, p$, the complexity incurred due to reorthogonalization is $O(p^3)$. The total runtime complexity of the algorithm is therefore $O(Np^2 + p^3)$. As for memory requirements, the algorithm constructs a basis $\boldsymbol{V}_p \in \mathbb{R}^{p \times p}$ and as such its memory complexity is $O(p^2)$. Since p is in the order of magnitude of millions, both time and memory complexity make SlowLanczos impractical.

Algorithm 2: FastLanczos($\boldsymbol{A}, M$)

Input: Linear operator $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ with spectrum in the range $[-1, 1]$.
Number of iterations M.
Result: Eigenvalues and eigenvectors of the tridiagonal matrix $\boldsymbol{T}_m$.
for $m = 1, \ldots, M$ **do**
  if $m == 1$ **then**
    sample $\boldsymbol{v} \sim \mathcal{N}(0, \boldsymbol{I})$;
    $\boldsymbol{v} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$;
    $\boldsymbol{v}_{\text{next}} = \boldsymbol{A}\boldsymbol{v}$;
  else
    $\boldsymbol{v}_{\text{next}} = \boldsymbol{A}\boldsymbol{v} - \beta_{m-1}\boldsymbol{v}_{\text{prev}}$;
  end
  $\alpha_m = \boldsymbol{v}_{\text{next}}^\top \boldsymbol{v}$;
  $\boldsymbol{v}_{\text{next}} = \boldsymbol{v}_{\text{next}} - \alpha_m \boldsymbol{v}$;
  $\beta_m = \|\boldsymbol{v}_{\text{next}}\|_2$;
  $\boldsymbol{v}_{\text{next}} = \frac{\boldsymbol{v}_{\text{next}}}{\beta_m}$;
  $\boldsymbol{v}_{\text{prev}} = \boldsymbol{v}$;
  $\boldsymbol{v} = \boldsymbol{v}_{\text{next}}$;
end

$$\boldsymbol{T}_M = \begin{bmatrix} \alpha_1, & \beta_1, & & & \\ \beta_1, & \alpha_2, & \beta_2, & & \\ & \beta_2, & \alpha_3, & & \\ & & & \ddots & \beta_{M-1} \\ & & & \beta_{M-1} & \alpha_M \end{bmatrix};$$

$\{\theta_m\}_{m=1}^M, \{\boldsymbol{y}_m\}_{m=1}^M = \text{eig}(\boldsymbol{T}_M)$;
return $\{\theta_m\}_{m=1}^M, \{\boldsymbol{y}_m\}_{m=1}^M$;

C.3 Spectral density estimation via FastLanczos

As a first step towards making Lanczos suitable for the problem we attack in this paper, we remove the reorthogonalization step in Algorithm 1. This allows us to save only three terms–$\boldsymbol{v}_{\text{prev}}$, $\boldsymbol{v}$ and $\boldsymbol{v}_{\text{next}}$–instead of the whole matrix $\boldsymbol{V}_m \in \mathbb{R}^{p \times m}$ (see Algorithm 2). This greatly reduces the memory complexity of the algorithm at the cost of a nuisance that is discussed in Section C.5. Moreover, it removes the $O(p^3)$ term from the runtime complexity.

In light of the runtime complexity analysis in the previous subsection, it is clear that running Lanczos for p iterations is impractical. Realizing that, the authors of Lin et al. (2016) proposed to run the algorithm for $M \ll p$ iterations and compute an approximation

Algorithm 3: LanczosApproxSpec
$(\boldsymbol{A}, M, K, n_{\text{vec}}, \kappa)$

Input: Linear operator $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ with spectrum in the range $[-1, 1]$.
Number of iterations M.
Number of points K.
Number of repetitions n_{vec}.

Result: Density of the spectrum of $\boldsymbol{A}$ evaluated at K evenly distributed points in the range $[-1, 1]$.

for $l = 1, \ldots, n_{vec}$ **do**
  $\{\theta_m^l\}_{m=1}^M, \{\boldsymbol{y}_m^l\}_{m=1}^M = \text{FastLanczos}(\boldsymbol{A}, M)$;
end
$\{t_k\}_{k=1}^K = \text{linspace}(-1, 1, K)$;
for $k = 1, \ldots, K$ **do**
  $\sigma = \frac{2}{(M-1)\sqrt{8\log(\kappa)}}$;
  $\phi_k = \frac{1}{n_{\text{vec}}} \sum_{l=1}^{n_{\text{vec}}} \sum_{m=1}^{M} y_m^l[1]^2 g_\sigma(t - \theta_m^l)$
end
return $\{\phi_k\}_{k=1}^K$;

to the spectrum based on the eigenvalues $\{\theta_m\}_{m=1}^M$ and eigenvectors $\{\boldsymbol{y}_m\}_{m=1}^M$ of $\boldsymbol{T}_M$. Denoting by $y_m[1]$ the first element in $\boldsymbol{y}_m$, their proposed approximation was $\hat{\phi}(t) = \sum_{m=1}^M y_m[1]^2 g_\sigma(t - \theta_m^l)$, where $g_\sigma(t - \theta_m^l)$ is a Gaussian with width σ centered at θ_m^l. Intuitively, instead of computing the true spectrum $\phi(t) = \frac{1}{p}\sum_{i=1}^p \delta(t - \lambda_i)$, their algorithm computes only $M \ll p$ eigenvalues and replaces each with a Gaussian bump. They further proposed to improve the approximation by starting the algorithm from several different starting vectors, $\boldsymbol{v}_1^l, l = 1, \ldots, n_{\text{vec}}$, and averaging the results. We summarize FastLanczos in Algorithm 2 and LanczosApproxSpec in Algorithm 3.

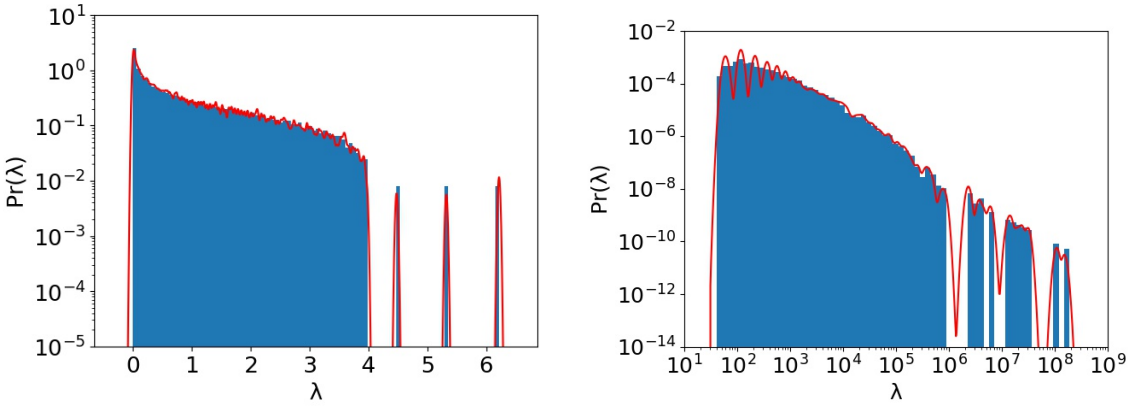
C.4 Complexity of FastLanczos

Each of the M iterations requires a single Hessian-vector multiplication. As previously mentioned, this product requires $O(Np)$ complexity. The total runtime complexity of the algorithm is therefore $O(MNp)$ (for $n_{vec} = 1$). Although the complexity might seem equivalent to that of training a model from scratch for M epochs, this is not the case. The batch size used for training a model is usually limited (128 in our case) so as to no deteriorate the model's generalization. Such limitations do not apply to FastLanczos, which can utilize the largest possible batch size that fits into the GPU memory (1024 in our case). As for memory requirements; we only save three vectors and as such the memory complexity is merely $O(p)$.

C.5 Reorthogonalization

Under exact arithmetic, the Lanczos algorithm constructs an orthonormal basis. However, in practice the calculations are performed in floating point arithmetic, resulting in loss of orthogonality. This is why the reorthogonalization step in Algorithm 1 was introduced in the

first place. From our experience, we did not find the lack of reorthogonalization to cause any issue, except for the known phenomenon of appearance of "ghost" eigenvalues–multiple copies of eigenvalues, which are unrelated to the actual multiplicities of the eigenvalues. Despite these, in all the toy examples we ran on synthetic data, we found that our method approximates the spectrum well, as is shown in Figure 15.



(a) **Verification of FastLanczos.** Approximating the spectrum of a matrix $\boldsymbol{Y} \in \mathbb{R}^{2000\times 2000}$, sampled from the distribution $\boldsymbol{Y} = \boldsymbol{X} + \frac{1}{2000}\boldsymbol{Z}\boldsymbol{Z}^\top$, where $X_{1,1} = 5$, $X_{2,2} = 4$, $X_{3,3} = 3$, $X_{i,j} = 0$ elsewhere and the entries of $\boldsymbol{Z} \in \mathbb{R}^{2000\times 2000}$ are standard normally distributed.

(b) **Verification of FastLanczos for approximating the log-spectrum.** Approximating the log-spectrum of a matrix $\boldsymbol{Y} \in \mathbb{R}^{1000\times 1000}$, sampled from the distribution $\boldsymbol{Y} = \frac{1}{1000}\boldsymbol{Z}\boldsymbol{Z}^\top$, where the entries of $\boldsymbol{Z} \in \mathbb{R}^{500\times 1000}$ are distributed i.i.d Pareto with index $\alpha = 1$. This type of matrices are known to have a power law spectral density.

Figure 15: **Verification of spectrum approximation on synthetic data.** The eigenvalues obtained from eigenvalue decomposition are plotted in blue color in a histogram with 100 bins. Our spectral approximation is plotted on top as a red line. In both the left and the right plots we average over $n_{\text{vec}} = 10$ initial vectors.

C.6 Normalization

As a first step towards approximating the spectrum of a large matrix, we renormalize its range to $[-1, 1]$. This can be done using any method that allows to approximate the maximal and minimal eigenvalue of a matrix–for example, the power method. In this work we follow the method proposed in Lin et al. (2016). This normalization has the benefit of allowing us to set σ to a fixed number, which does not depend on the specific spectrum approximated. We summarize the procedure in Algorithm 4.

C.7 Spectral density estimation of $f(A)$

Figure 16a approximates the spectrum of $\boldsymbol{E}$, showing that it is approximately linear on a log-log plot. A better idea would have been to approximate the spectrum of $\log(|\boldsymbol{E}|)$ in the first place (the absolute value is due to $\boldsymbol{E}$ being symmetric about the origin), since this would lead to a more precise estimate. Mathematically speaking, this amounts to approximating

Algorithm 4: Normalization$(\boldsymbol{A}, M_0, \tau)$

Input: Linear operator $\boldsymbol{A} \in \mathbb{R}^{p \times p}$.
Number of iterations M_0.
Margin percentage τ.
Result: Linear operator $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ with spectrum in the range $[-1, 1]$.
`/* approximate minimal and maximal eigenvalues */`
$\{\theta_m\}_{m=1}^{M_0}, \{\boldsymbol{y}_m\}_{m=1}^{M_0} = \text{FastLanczos}(\boldsymbol{A}, M_0)$;
$\lambda_{\min} = \theta_1 - \|(\boldsymbol{A} - \theta_1 \boldsymbol{I})\boldsymbol{y}_1\|$;
$\lambda_{\max} = \theta_{M_0} + \|(\boldsymbol{A} - \theta_{M_0} \boldsymbol{I})y_{M_0}\|$;
`/* add margin */`
$\Delta = \tau(\lambda_{\max} - \lambda_{\min})$;
$\lambda_{\min} = \lambda_{\min} - \Delta$;
$\lambda_{\max} = \lambda_{\max} + \Delta$;
`/* normalized operator */`
$c = \frac{\lambda_{\min} + \lambda_{\max}}{2}$;
$d = \frac{\lambda_{\max} - \lambda_{\min}}{2}$;
return $\frac{\boldsymbol{A} - c\boldsymbol{I}}{d}$;

the measure $\Pr(\log(|\lambda|))d\log(|\lambda|)$ instead of $\Pr(\lambda)d\lambda$. Using change of measure arguments, we have

$$\Pr(\log(|\lambda|))d\log(|\lambda|) = \Pr(\log(|\lambda|))\frac{d\log(|\lambda|)}{d|\lambda|}d|\lambda| = \Pr(\log(|\lambda|))\frac{1}{|\lambda|}d|\lambda|.$$

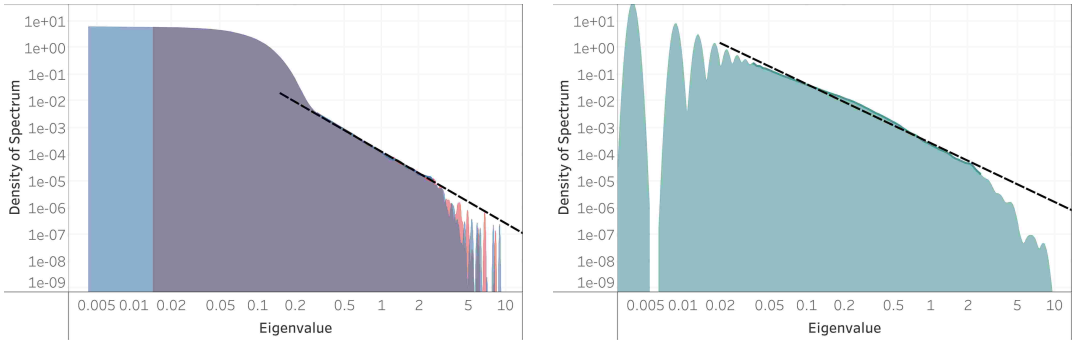
Following the ideas presented in Section C.3, the above can be approximated as

$$\sum_{m=1}^{M} y_m[1]^2 \frac{1}{\theta_m^l} g_\sigma(|\lambda| - \log\left(\theta_m^l\right)).$$

Implementation-wise, all that is required is to replace θ_m^l with $\log\left(\theta_m^l\right)$ in Algorithm 3, scale the Gaussian bumps by $\frac{1}{\theta_m^l}$, and to apply log on $|\lambda_{\min}|$ and $|\lambda_{\max}|$ before adding the margin in Algorithm 4. In practice, we apply $f = \log(|\lambda| + \epsilon)$, where ϵ is a small constant added for numerical stability. Figure 16b shows the outcome of such procedure. The idea of approximating the log of the spectrum (or any function of it) is inspired by a recent work of Ubaru et al. (2017), which suggests a method for approximating $\text{Tr}\{f(\boldsymbol{A})\}$ using Lanczos and comments that similar ideas could be used for approximating functions of matrix spectra.

C.8 SubspaceIteration

The spectrum of the Hessian follows a bulk-and-outliers structure. Moreover, the number of outliers is approximately equal to C, the number of classes in the classification problem. It is therefore natural to extract the top C outliers using, for example, the subspace iteration algorithm, and then to apply LanczosApproxSpec on a rank C deflated operator to approximate the bulk. We demonstrate the benefits of SubspaceIteration in Figure 17 and summarize its steps in Algorithm 5. The runtime complexity of SubspaceIteration



(a) Spectrum of $\boldsymbol{E}$ on a logarithmic x-axis scale

(b) Spectrum of $\log(\boldsymbol{E})$

Figure 16: **Tail properties of $\boldsymbol{E}$.** Spectrum of the test $\boldsymbol{E}$ for VGG11 trained on MNIST sub-sampled to 1351 examples per class. On the left we approximate the spectrum of $\boldsymbol{E}$ and plot it on a logarithmic x-axis. The positive eigenvalues of $\boldsymbol{E}$ are plotted in red and the absolute value of the negative ones in blue. Notice how the spectrum is almost perfectly symmetric about the origin. Fitting a power law trend on part of the spectrum results in a fit $\phi = 1.2{\times}10^{-4}\ |\lambda|^{-2.7}$ with an R^2 of 0.99. On the right we approximate the spectrum of $\log(\boldsymbol{E})$. Fitting a power law trend results in a fit $\phi = 2.6{\times}10^{-4}\ |\lambda|^{-2.2}$ with an R^2 of 0.99. These spectra can not originate from Wigner's semicircle law, nor other classical RMT distributions

Algorithm 5: SUBSPACEITERATION$(\boldsymbol{A}, C, T)$

Input: Linear operator $\boldsymbol{A} \in \mathbb{R}^{p\times p}$.
Rank C.
Number of iterations T.
Result: Eigenvalues $\{\lambda_c\}_{c=1}^{C}$.
Eigenvectors $\{\boldsymbol{v}_c\}_{c=1}^{C}$.

```
for c = 1, ..., C do
    sample v_c ~ N(0, I);
    v_c = v_c / ||v_c||_2;
end
Q = QR(V);
for t = 1, ..., T do
    V = AQ;
    Q = QR(V);
end
for c = 1, ..., C do
    λ_c = ||v_c||_2
end
return {λ_c}_{c=1}^C, {v_c}_{c=1}^C
```

is $O(TC^2Np)$, T being the number of iterations, which is C^2 times higher than that of FASTLANCZOS.

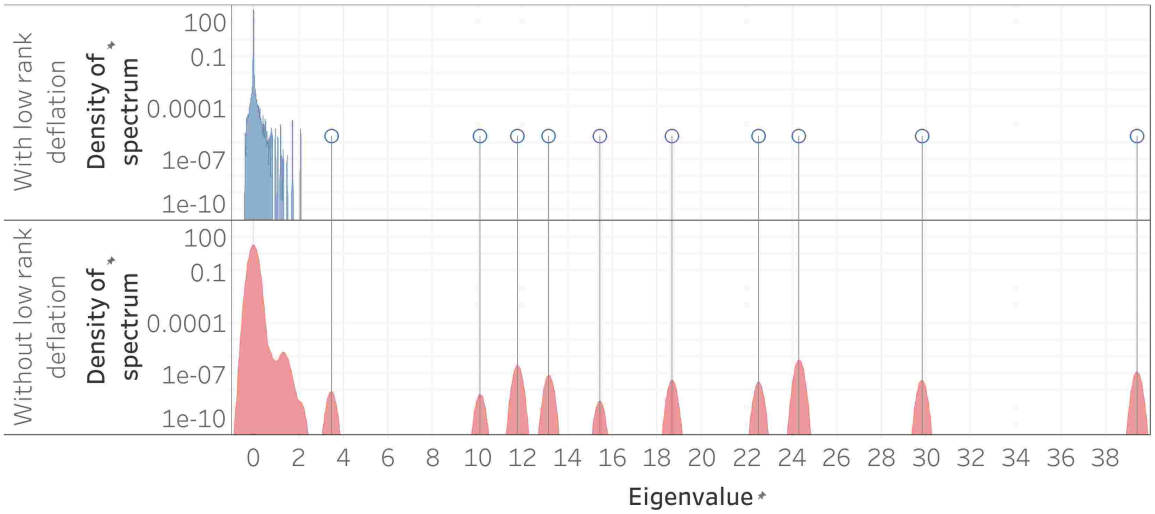


Figure 17: **Benefits of low rank deflation using subspace iteration.** Spectrum of the train Hessian for ResNet18 trained on MNIST with 136 examples per class. Top panel: SubspaceIteration followed by LanczosApproxSpec. Bottom panel: LanczosApproxSpec only. Notice how the top eigenvalues at the top panel align with the outliers in the bottom panel. Low rank deflation allows for precise detection of outlier location and improved 'resolution' of the bulk distribution.

Appendix D. Experimental details

D.1 Training networks

We present here results from training the VGG11 (Simonyan and Zisserman, 2014) and ResNet18 (He et al., 2016) architectures on the MNIST (LeCun et al., 2010), FashionMNIST (Xiao et al., 2017), CIFAR10 and CIFAR100 (Krizhevsky and Hinton, 2009) datasets. We use stochastic gradient descent with 0.9 momentum, 5×10^{-4} weight decay and 128 batch size. We train for 200 epochs in the case of MNIST and FashionMNIST and 350 in the case of CIFAR10 and CIFAR100, annealing the initial learning rate by a factor of 10 at 1/3 and 2/3 of the number of epochs. For each dataset and network, we sweep over 100 logarithmically spaced initial learning rates in the range $[0.25, 0.0001]$ and pick the one that results in the best test error in the last epoch. For each dataset and network, we repeat the previous experiments on 20 training sample sizes logarithmically spaced in the range $[10, 5000]$.

We also train an eight-layer multilayer perceptrons (MLP) with 2048 neurons in each hidden layer on the same datasets. We use the same hyperparameters, except we train for 350 epochs for all datasets and optimize the initial learning rate over 25 logarithmically spaced values.

The massive computational experiments reported here were run painlessly using ClusterJob and ElastiCluster (Monajemi and Donoho, 2015; Monajemi et al., 2017, 2019).

D.2 Analyzing the spectra

For each operator $\boldsymbol{A}$, we begin by computing Normalization($\boldsymbol{A}, M_0$=32, τ=0.05). We then approximate the spectrum using LanczosApproxSpec(A, M, K=1024, n_{vec}=1, κ=3), where $M \in \{128, 256\}$. Finally, we denormalize the spectrum into its original range (which is not $[-1, 1]$). Optionally, we apply the above steps on a rank-C deflated operator obtained using
SubspaceIteration($\boldsymbol{A}, C, T$=128). Optionally, we compute the log-spectrum, in which case we use M=2048 iterations and a different value of κ.

D.3 Removing sources of randomness

The methods we employ in this paper–including Lanczos and subspace iteration–assume deterministic linear operators. As such, we train our networks without preprocessing the input data using random flips or crops. Moreover, we replace dropout layers (Srivastava et al., 2014) with batch normalization ones (Ioffe and Szegedy, 2015) in the VGG architecture. The batch normalization layers are always set to "test mode".

References

Guillaume Alain, Nicolas Le Roux, and Pierre-Antoine Manzagol. Negative eigenvalues of the hessian in deep neural networks. *arXiv preprint arXiv:1902.02366*, 2019.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276, 1998.

Anders Andreassen and Ethan Dyer. Asymptotics of wide convolutional neural networks. *arXiv preprint arXiv:2008.08675*, 2020.

Jimmy Ba, Roger Grosse, and James Martens. Distributed second-order optimization using kronecker-factored approximations. 2016.

Dankmar Böhning. Multinomial logistic regression algorithm. *Annals of the institute of Statistical Mathematics*, 44(1):197–200, 1992.

Avraam Chatzimichailidis, Franz-Josef Pfreundt, Nicolas R Gauger, and Janis Keuper. Gradvis: Visualization and second order analysis of optimization surfaces during the training of deep neural networks. *arXiv preprint arXiv:1909.12108*, 2019.

Ronan Collobert and Samy Bengio. A gentle hessian for efficient gradient descent. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages V–517. IEEE, 2004.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

David A Drabold and Otto F Sankey. Maximum entropy approach for linear scaling in the electronic structure problem. *Physical review letters*, 70(23):3631, 1993.

Francois Ducastelle and Françoise Cyrot-Lackmann. Moments developments and their application to the electronic charge distribution of d bands. *Journal of physics and chemistry of solids*, 31(6):1295–1306, 1970.

Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Stanislav Fort and Surya Ganguli. Emergent properties of the local geometry of neural loss landscapes. *arXiv preprint arXiv:1910.05929*, 2019.

Stanislav Fort and Stanisław Jastrzębski. Large scale structure of neural network loss landscapes. *arXiv preprint arXiv:1906.04724*, 2019.

Stanislav Fort and Adam Scherlis. The goldilocks zone: Towards better understanding of neural network loss landscapes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3574–3581, 2019.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *arXiv preprint arXiv:1906.08034*, 2019.

Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, 2020.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241, 2019.

Diego Granziol. Curvature is key: Sub-sampled loss surfaces and the implications for large batch training. *arXiv preprint arXiv:2006.09092*, 2020.

Diego Granziol, Timur Garipov, Stefan Zohren, Dmitry Vetrov, Stephen Roberts, and Andrew Gordon Wilson. The deep learning limit: are negative neural network eigenvalues just noise?

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582, 2016.

Roger Grosse and Ruslan Salakhudinov. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. In *International Conference on Machine Learning*, pages 2304–2313, 2015.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Carl J Huberty and Stephen Olejnik. *Applied MANOVA and discriminant analysis*, volume 498. John Wiley & Sons, 2006.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. The asymptotic spectrum of the hessian of dnn throughout training. *arXiv preprint arXiv:1910.02875*, 2019a.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Freeze and chaos for dnns: an ntk view of batch normalization, checkerboard and boundary effects. *arXiv preprint arXiv:1907.05715*, 2019b.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. *arXiv preprint arXiv:1807.05031*, 2018a.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. 2018b.

Stanisław Jastrzębski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on optimization trajectories of deep neural networks. *arXiv preprint arXiv:2002.09572*, 2020.

Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. In *International Conference on Learning Representations*, 2018.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Pathological spectra of the fisher information metric and its variants in deep neural networks. *arXiv preprint arXiv:1910.05992*, 2019a.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: Mean field approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1032–1041, 2019b.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Sudhir Kylasa, Fred Roosta, Michael W Mahoney, and Ananth Grama. Gpu accelerated sub-sampled newton's method for convex classification problems. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 702–710. SIAM, 2019.

Cornelius Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

Y LeCun, L Bottou, G Orr, and K Muller. Efficient backprop in neural networks: Tricks of the trade (orr, g. and müller, k., eds.). *Lecture Notes in Computer Science*, 1524(98):111, 1998.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2:18, 2010.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

Xinyan Li, Qilong Gu, Yingxue Zhou, Tiancong Chen, and Arindam Banerjee. Hessian based analysis of sgd for deep nets: Dynamics and generalization. *arXiv preprint arXiv:1907.10732*, 2019.

Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM review*, 58(1):34–65, 2016.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

Michael Mahoney and Charles Martin. Traditional and heavy tailed self regularization in neural network models. In *International Conference on Machine Learning*, pages 4284–4293, 2019.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *arXiv preprint arXiv:1810.01075*, 2018.

H. Monajemi and D. L. Donoho. Clusterjob: An automated system for painless and reproducible massive computational experiments. `https://github.com/monajemi/clusterjob`, 2015.

H. Monajemi, D. L. Donoho, and V. Stodden. Making massive computational experiments painless. *Big Data (Big Data), 2016 IEEE International Conference on*, February 2017.

H. Monajemi, R. Murri, E. Yonas, P. Liang, V. Stodden, and D.L. Donoho. Ambitious data science can be painless. *arXiv:1901.08705*, 2019.

Samet Oymak, Zalan Fabian, Mingchen Li, and Mahdi Soltanolkotabi. Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian. *arXiv preprint arXiv:1906.05392*, 2019.

Vardan Papyan. The full spectrum of deep net hessians at scale: Dynamics with sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In *International Conference on Machine Learning*, pages 5012–5021, 2019.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Jeffrey Pennington and Yasaman Bahri. Geometry of neural network loss surfaces via random matrix theory. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2798–2806. JMLR. org, 2017.

Jeffrey Pennington and Pratik Worah. The spectrum of the fisher information matrix of a single-hidden-layer neural network. In *Advances in Neural Information Processing Systems*, pages 5410–5419, 2018.

Lukas Pfahler and Katharina Morik. Evolution of eigenvalue decay in deep networks. 2019.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

I Turek. A maximum-entropy approach to the density of states within the recursion method. *Journal of Physics C: Solid State Physics*, 21(17):3251, 1988.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Charles F Van Loan and Nikos Pitsianis. Approximation with kronecker products. In *Linear algebra for large scale and real-time applications*, pages 293–314. Springer, 1993.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. *arXiv preprint arXiv:1806.05236*, 2018.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. *arXiv preprint arXiv:1905.05934*, 2019.

Jialei Wang and Tong Zhang. Utilizing second order information in minibatch stochastic variance reduced proximal iterations. *J. Mach. Learn. Res.*, 20:42–1, 2019.

John C Wheeler and Carl Blumstein. Modified moments for harmonic solids. *Physical Review B*, 6(12):4380, 1972.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017a.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Second-order optimization for deep reinforcement learning using kronecker-factored approximation. In *NIPS*, 2017b.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Peng Xu, Fred Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, pages 1–36, 2019.

Peng Xu, Fred Roosta, and Michael W Mahoney. Second-order optimization for non-convex machine learning: An empirical study. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 199–207. SIAM, 2020.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael Mahoney. Pyhessian: Neural networks through the lens of the hessian. *arXiv preprint arXiv:1912.07145*, 2019.

Haishan Ye, Zhichao Huang, Cong Fang, Chris Junchi Li, and Tong Zhang. Hessian-aware zeroth-order optimization for black-box adversarial attack. *arXiv preprint arXiv:1812.11377*, 2018.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *arXiv preprint arXiv:1902.01996*, 2019.