EMPIRICAL ANALYSIS OF THE HESSIAN OF OVER-PARAMETRIZED NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the properties of common loss surfaces through their Hessian matrix. In particular, in the context of deep learning, we empirically show that the spectrum of the Hessian is composed of two parts: (1) the bulk centered near zero, (2) and outliers away from the bulk. We present numerical evidence and mathematical justifications to the following conjectures laid out by Sagun et al. (2016): Fixing data, increasing the number of parameters merely scales the bulk of the spectrum; fixing the dimension and changing the data (for instance adding more clusters or making the data less separable) only affects the outliers. We believe that our observations have striking implications for non-convex optimization in high dimensions. First, the flatness of such landscapes (which can be measured by the singularity of the Hessian) implies that classical notions of basins of attraction may be quite misleading. And that the discussion of wide/narrow basins may be in need of a new perspective around over-parametrization and redundancy that are able to create large connected components at the bottom of the landscape. Second, the dependence of small number of large eigenvalues to the data distribution can be linked to the spectrum of the covariance matrix of gradients of model outputs. With this in mind, we may reevaluate the connections within the data-architecture-algorithm framework of a model, hoping that it would shed light into the geometry of high-dimensional and non-convex spaces in modern applications. In particular, we present a case that links the two observations: small and large batch gradient descent appear to converge to different basins of attraction but we show that they are in fact connected through their flat region and so belong to the same basin.

1 INTRODUCTION

In this paper we study the geometry of the loss surface of supervised learning problems through the lens of their second order properties. To introduce the framework, suppose we are given data in the form of input-label pairs, \( D = \{ (x^i, y^i) \}_{i=1}^N \) where \( x \in \mathbb{R}^d \) and \( y \in \mathbb{R} \) that are sampled i.i.d. from a possibly unknown distribution \( \nu \), a model that is parametrized by \( w \in \mathbb{R}^M \); so that the number of examples is \( N \) and the number of parameters of the system is \( M \). Suppose also that there is a predictor \( f(w, x) \). The supervised learning process aims to solve for \( w \) so that \( f(w, x) \approx y \). To make the ‘\( \approx \)’ precise, we use a non-negative loss function that measures how close the predictor is to the true label, \( \ell(f(w, x), y) \). We wish to find a parameter \( w^\ast \) such that:

\[
  w^\ast = \text{arg min } \mathcal{L}(w),
\]

where

\[
  \mathcal{L}(w) := \frac{1}{N} \sum_{i=1}^N \ell(f(w, x^i), y^i).
\]

In particular, one is curious about the relationship between \( \mathcal{L}(w) \) and \( \mathcal{L}'(w) := \int \ell d(\nu) \). By the law of large numbers, at a given point \( w \), \( \mathcal{L}_w \to \mathcal{L}'_w \) almost surely as \( N \to \infty \) for fixed \( M \). However in modern applications, especially in deep learning, the number of parameters \( M \) is comparable to the number of examples \( N \) (if not much larger). And the behaviour of the two quantities may be drastically different (for a recent analysis on provable estimates see Mei et al. (2016)).
A classical algorithm to find \( w^\star \) is gradient descent (GD), in which the optimization process is carried out using the gradient of \( \mathcal{L} \). A new parameter is found iteratively by taking a step in the direction of the negative gradient whose size is scaled with a constant step size \( \eta \) that is chosen from line-search minimization. Two problems emerge: (1) Gradient computation can be expensive, (2) Line-search can be expensive. More involved algorithms, such as Newton type methods, make use of second order information \(^{(\text{Nocedal & Wright, 2006})} \). Under sufficient regularity conditions we may observe:

\[
\mathcal{L}(w + \Delta w) \approx \mathcal{L}(w) + \Delta w^T \nabla \mathcal{L}(w) + \Delta w^T \nabla^2 \mathcal{L}(w) \Delta w.
\]

A third problem emerges beyond an even more expansive computational cost of the Hessian: (3) Most methods require the Hessian to be non-degenerate to a certain extent.

When the gradients are computationally expensive, one can alternatively use it’s stochastic version (SGD) that replaces the above gradient with the gradient of averages of losses over subsets such that \( \eta \) is chosen from line-search \(^{\text{(Bottou, 2010)}} \). The benefit of SGD on real-life time limits is obvious, and GD may be impractical for practical purposes in many problems. In any case, the stochastic gradient can be seen as an approximation to the true gradient, and hence it is important to understand how the two directions are related in the parameter space. Therefore, the discussion around the geometry of the loss surface can be enlightening in the comparison of the two algorithms: Does SGD locate solutions of a different nature than GD? Do they follow different paths? If so, which one is better in terms of generalization performance?

For the second problem of expensive line-search, there are two classical solutions: using a small, constant step size, or scheduling the step size according to a certain rule. In practice, in the context of deep learning, the values for both approaches are determined heuristically, by trial and error. More involved optimal step size choices involve some kind of second order information that can be obtained from the Hessian of the loss function \(^{\text{(Schaul et al., 2013)}} \). From a computational point of view, obtaining the Hessian is extremely expensive, however obtaining some of its largest and smallest eigenvalues and eigenvectors are not that expensive. Is it enough to know only those eigenvalues and eigenvectors that are large in magnitude? How do they change through training? Would such a method work in SGD as well as it would on GD?

For the third problem, let’s look at the Hessian a little closer. A critical point is defined by \( w \) such that \( \| \nabla \mathcal{L}(w) \| = 0 \) and the nature of it can be determined by looking at the signs of its Hessian matrix. If all eigenvalues are positive the point is called a local minimum, if \( r \) of them are negative and the rest are positive, then it is called a saddle point with index \( r \). At the critical point, the eigenvectors indicate the directions in which the value of the function locally changes. Moreover the changes are proportional to the corresponding signed eigenvalue. Under sufficient regularity conditions, it is rather straightforward to show that gradient based methods converge to points where gradient is zero. Recently \(^{\text{(Lee et al., 2016)}} \) showed that they indeed converge to minimizers. However, a significant and untested assumption to establish these convergence results is that the loss is non-degenerate. A relaxation of the above convergence to the case of non-isolated critical points can be found in \(^{\text{(Panageas & Piliouras, 2016)}} \). What about the critical points of machine learning loss functions? Do they satisfy the non-degeneracy assumptions? If they don’t, can we still apply the results of provable theorems to gain intuition?

### 1.1 A HISTORICAL OVERVIEW

One of the first instances of the comparison of GD and SGD in the context of neural networks dates back to late eighties and early nineties. \(^{\text{(Bottou, 1991)}} \) points out that large eigenvalues of the Hessian of the loss can create the illusion of a local minima and GD can get stuck there, it further claims that the help of the inherent noise in SGD may help getting out of this obstacle. The origin of this observation is due to \(^{\text{(Bourely, 1989)}} \), as well as numerical justifications. However, given the computational limits of the time, these experiments relied on low-dimensional neural networks with few hidden units. The picture may be drastically different in higher dimensions. In fact, provable results in statistical physics tell us that, in certain real-valued non-convex functions, the local minima concentrate at an error level near that of the global minima. A theoretical review on this can be found in \(^{\text{(Auffinger et al., 2013)}} \), while \(^{\text{(Sagun et al., 2014)}} \) and \(^{\text{(Ballard et al., 2017)}} \) provide an experimental simulation as well as a numerical study for neural networks. They notably find that high error local minima traps do not appear when the model is over-parametrized.
These concentration results can help explain why we find that the solutions attained by different optimizers like GD and SGD often have comparable training accuracies. However, while these methods find comparable solutions in terms of training error there is no guarantee they generalize equally. A recent work in this direction compares the generalization performance of small batch and large batch methods (Keskar et al., 2016). They demonstrate that the large batch methods always generalize a little bit worse even when they have similar training accuracies. The paper further makes the observation that the basins found by small batch methods are wider, thereby contributing to the claim that wide basins, as opposed to narrow ones, generalize better.

The final part of the historical account is devoted to the observation of flatness of the landscape in neural networks and its consequences through the lens of the Hessian. In early nineties, Hochreiter & Schmidhuber (1997) remarks that there are parts of the landscape in which the weights can be perturbed without significantly changing the loss value. Such regions at the bottom of the landscape are called the flat minima, which can be considered as another way of saying a very wide minima. It is further noted that such minima have better generalization properties and a new loss function that makes use of the Hessian of the loss function has been proposed that targets the flat minima. The computational complexity issues have been attempted to be resolved using the $R$-operator of Pearlmutter (1994). However, the new loss requires all the entries of the Hessian, and even with the $R$-operator it is unimaginably slow for today’s large networks. More recently, an exact numerical calculation of the Hessian have been carried out by Sagun et al. (2016). It turns out that the Hessian is degenerate at any given point including the randomly chosen initial point, that the spectrum of it is composed of two parts: (1) the bulk, and (2) the outliers. The bulk is mostly full of zero eigenvalues with a fast decaying tail, and the outliers are only a handful which appears to depend on the data. This implies that, locally, most directions in the weight space are flat, and leads to little or no change in the loss value, except for the directions of eigenvectors that correspond to the large eigenvalues of the Hessian.

In this work, we study the problems of the third class and discuss their impact on the first two.

1.2 OVERVIEW OF RESULTS

In an attempt to develop an understanding for some of the classical and modern problems described above and hopefully to get further insight into non-convex optimization over high dimensional spaces, we study the loss function through its second order properties through the spectrum of its Hessian matrix.

---

1 A general remark is that the terms wide-narrow, high-low, are all relative. In a recent study, Dinh et al. (2017) shows how sharp minima can still generalize with proper modifications to the loss function. We note that it takes a non-linear transformation to deform relative widths of basins. And, in this work, we focus on relative values as opposed to absolute values to get a consistent comparison across different setups.
We study the Hessian using a decomposition of it as a sum of two matrices, where the first one is

\[ H = \nabla^2 \ell(\hat{w}) \]

with a real valued output function, and classification: the negative log likelihood loss composed within the data-architecture-algorithm triangle. We begin by discussing the decomposition of the Hessian.

The first observation is that the Hessian is not slightly singular but extremely so, having almost all of its eigenvalues at or near zero. And this observations seems to hold even at random points of the space. For example, let us consider an artificial neural network with two layers (totaling $5K$ parameters) and trained using gradient descent. Figure [1] shows the full spectrum of the Hessian at the random initial point of training and after the final training point. Intuitively, this kind of singularity should be expected to emerge if (1) the data is essentially low dimensional but lives in a larger ambient space, and (2) the model is over-parametrized. In the context of deep learning, over-parametrization is a key feature of the model, and in most cases the data is expected to be essentially on a low dimensional manifold.

We study the Hessian using a decomposition of it as a sum of two matrices, where the first one is the sample covariance matrix of the gradients of model outputs and the second one is the Hessian of the function that describes the model outputs. We argue that the second term can be ignored as training progresses which leaves us with the covariance term. This connection to the covariance of the gradients is then used to explain why having more parameters than samples results in degenerate Hessian matrices. Finally, we empirically examine the spectrum to uncover intricate dependencies within the data-architecture-algorithm triangle. We begin by discussing the decomposition of the Hessian.

2 Generalized Gauss-Newton decomposition of the Hessian

In order to study its spectrum, we will describe how the Hessian can be decomposed into two meaningful matrices [LeCun et al., 1998; Martens, 2010]. Suppose the loss function is given as a composition of two functions, the model function $f: \mathbb{R}^M \rightarrow \mathbb{R}$ is the real valued output of a network that depends on the parameters; and the loss function $\ell: \mathbb{R} \rightarrow \mathbb{R}^+$ is a convex function. Here, $\bullet$ refers to the given example. Examples include the regression: the mean-square loss composed with a real valued output function, and classification: the negative log likelihood loss composed with the dot product of the output of a softmax layer with the label vector.

For ease of reading, we indicate the dependencies of functions $\ell$ and $f$ to data by the index of the example, or omit it altogether in case it is not necessary, and unless noted otherwise the gradients are taken with respect to $w$. The gradient and the Hessian of the loss for for a given example are given by

\[
\nabla \ell(f(w)) = \ell'(f(w))\nabla f(w) \quad (3)
\]

\[
\nabla^2 \ell(f(w)) = \ell''(f(w))\nabla f(w)\nabla f(w)^T + \ell'(f(w))\nabla^2 f(w) \quad (4)
\]

where $\bullet^T$ denotes the transpose operation (here the gradient is a column-vector). Note that since $\ell$ is convex $\ell''(s) \geq 0$ and we can take its square root which allows us to rewrite the Hessian of the loss as follows:

\[
\nabla^2 \ell(w) = \frac{1}{N} \sum_{i=1}^{N} \left[ \sqrt{\ell''(f_i(w))} \nabla f_i(w) \right] \left[ \sqrt{\ell''(f_i(w))} \nabla f_i(w)^T \right] + \frac{1}{N} \sum_{i=1}^{N} \ell'(f_i(w))\nabla^2 f_i(w) \quad (5)
\]

Next, we argue how we can ignore the second term in this decomposition.

2.1 Hessian at the bottom of the landscape

In general, it isn’t straightforward to discuss the spectrum of the sums of matrices by looking at the individual ones. Nevertheless, looking at the decomposition, we can still infer what we should expect. At a point close to a local minimum, the average gradient is close to zero. However, this doesn’t necessarily imply that the gradients for individual samples are also zero. However, if $\ell'(f(\hat{w}))$ and $\nabla f(\hat{w})$ are not correlated, then we can ignore the second term. And so the Hessian can be approximated by the first term:

\[
\nabla^2 \ell(\hat{w}) \approx \frac{1}{N} \sum_{i=1}^{N} \left[ \sqrt{\ell''(f_i(\hat{w}))} \nabla f_i(\hat{w}) \right] \left[ \sqrt{\ell''(f_i(\hat{w}))} \nabla f_i(\hat{w})^T \right] \quad (6)
\]
Here, Equation 6 is the sum of rank one matrices (via the outer products of gradients of $f$ multiplied by some non-negative number), therefore, the sum can be written as a product of an $M \times N$ matrix with its transpose where the columns of the matrix are formed by the scaled gradients of $f$. Immediately, this implies that there are at least $M - N$ many trivial eigenvalues of the Hessian.

3 IMPLICATIONS OF FLATNESS FOR THE GEOMETRY OF THE ENERGY LANDSCAPE

In this section we focus on implications of the above observations on training. We discuss how data, model, and algorithm affect the spectrum of the Hessian of the loss.

3.1 THE RELATION BETWEEN DATA AND EIGENVALUES

In many cases of practical interest, the data contains contains redundancies. In such cases, the number of trivial eigenvalues could be even more smaller than $N$. For instance, if one deals with a classification problem where the training data has $k$ classes with relatively small deviation among each of them, it is reasonable to expect that there will be an order of $k$ many non-trivial eigenvalues of the Hessian (assuming the approximation is valid).

To test this idea, we used a feed-forward neural network with a 100 dimensional input layer, two hidden layers each of which with 30 hidden units, and a $k$ dimensional output layer that is combined with softmax for $k$-class classification. We sampled random $k$ Gaussian clusters in the input space and normalized the data globally. Then we carried out the training using SGD on ReLU network for the following number of clusters: $k \in \{2, 5, 10, 20, 50\}$. We look at the data through the lens of principle component analysis (PCA), and next to it we calculate the spectrum of the Hessian of the loss function.

Even though the number of large principle components do not match the number of clusters exactly in Figure 2, the number of large eigenvalues that are above the gap in Figure 3 match exactly the number of classes in the dataset. Please refer to the table in the appendix for more experiments on this.

From a theoretical point of view, there are provable results that map the eigenvalues of the sample covariance matrix to the population matrix, a recent result can be found in Bloemendal et al. (2016). Some machine learning problems can be put into such a framework, however, results require independent inputs, and extensions to correlated data appears to be unavailable to the best of our knowledge. Please refer to the appendix for such a theoretical approach to this eigenvalue problem.
3.2 The relation between number of parameters and eigenvalues

We test the effect of growing the size of the network with fixed data, architecture, and algorithm. We sample 1K examples from the MNIST dataset, and train all the networks with the same step size until a sufficient stopping condition is satisfied. Then we compute the exact Hessian and plot its eigenvalues in order. In the figures, we show the left and right edges of the spectrum as the rest are just very close zero. Note, also, that for Figure 5 the x-axis indicates the ordered counts of eigenvalues, whereas for Figure 6 the x-axis is scaled to indicate the ratios of eigenvalues.

Our first observation is that there exist negative eigenvalues. This suggests that theoretical convergence results may be reached at time scales much larger than what is required to train a network. Also, the second immediate observation is that the magnitudes of negative eigenvalues (the left edge) are tiny compared to the ones at the positive eigenvalues (the right edge). Also note that, for the right edge of the spectrum (that is, for the large positive eigenvalues), the shape of the plot remains invariant as the number of parameters increase (Figure 4).

For the left edge of the spectrum, that is, for the small but negative eigenvalues, the shape of the plots of ordered eigenvalues shifts as the number of parameters increase (Figure 5). However, once we...
look at the ratios of the negative eigenvalues, we observe that increasing the number of dimensions keeps the ratio of small negative eigenvalues fixed. Moreover, they appear to be converging to have the same shape (Figure 6). This observation confirms our previous suspicions: (1) the effects of the ignored term in the decomposition is small (strictly speaking it’s contribution is small for the negative eigenvalues), (2) adding more nodes in the hidden layers proportionally enlarge the bulk, and (3) the large positive eigenvalues depend on data, so that their number should not, in principle, change with the increasing dimensionality of the parameter space.

3.3 The relation between the algorithm and eigenvalues

Finally, we turn to describing the nature of solutions found by the large and small batch methods for the training landscape. We train a convnet on MNIST using large and small batch methods, we find a learning rate for which both algorithms converge and use the same rate when training. Note that the stochastic gradients are averaged over the mini batch. Therefore, fixing the learning rate allows the algorithms to take steps whose lengths are proportional to the norms of the corresponding stochastic gradients averaged over the mini batch. This way, the variance of the steps are different, but the means are similar. We train until same number of iterations have been reached. Then, we calculate the Hessian and the spectrum of the Hessian (see Figures 7 and 8).

We observe that the negative eigenvalues are orders of magnitude smaller than the large ones, they also elbow at similar levels. More importantly, we observe that the number of large eigenvalues are the same for both methods, but their magnitude are different. The large batch method locates points with larger positive eigenvalues. This observation is consistent with [Keskar et al. 2016] as the way
they measure flatness takes calculates local rates of increase in a neighborhood of the solution which is intimately linked by the size of the large eigenvalues.

In the next and final section, we will argue that, contrary to what is believed in Keskar et al. (2016) and Chaudhari et al. (2016), the two algorithms do not have to be falling into different basins.

4 DISCUSSION ON BASINS OF SOLUTIONS AND GENERALIZATION

Finally, we revisit the issue through the lens of the following question: What does over-parametrization imply on the discussion around GD vs. SGD (or large batch vs small batch) especially for their generalization properties?

As noted in the introduction, for a while the common sense explanation on why SGD works well (in fact better) than GD (or large batch methods) was that the non-convex landscape had local minima at high energies which would trap large-batch or full-batch methods. Something that SGD with small batch shouldn’t suffer due to the inherent noise in the algorithm. However, there are various experiments that have been carried out in the past that show that, for reasonable large systems, this is not the case. For instance, Sagun et al. (2014) demonstrate that a two hidden layer fully connected network on MNIST can be trained by GD to reach at the same level of loss values as SGD. In fact, when the step size is fixed to the same value for both of the algorithms, they reach the same loss value at the same number of iterations. The training accuracy for both algorithms are the same, and the gap between test accuracies diminish as the size of the network increase with GD falling ever so slightly behind. It is also shown in Keskar et al. (2016) that training accuracies for both large and small batch methods are comparably good. Furthermore, Zhang et al. (2016) demonstrates that training landscape is easy to optimize even when there is no clear notion of generalization. Such observations are consistent with our observations: over-parametrization (due to the architecture of the model) leads to flatness at the bottom of the landscape which is easy to optimize.

When we turn our attention to generalization, Keskar et al. (2016) note that LB methods find a basin that is different than the one found by SB methods, and they are characterized by how wide the basin is. As noted in Figure 7, indeed the large eigenvalues are larger in LB than in SB, but is it enough to justify that they are in different basins, especially given the fact that the number of flat directions are enormous.

4.1 ARE THEY REALLY DIFFERENT BASINS? CASE OF CIFAR10 WITH LB VS SB

A common way to plot training profiles in larger scale neural networks is to stop every epoch to reserve extra computational power to calculate various statistics of the model at its current position. This becomes problematic when one compares training with different batch sizes, primarily because the larger batch model takes fewer steps in a given epoch. Recall that the overall loss is averaged, therefore, for a fixed point in the weight space, the empirical average of the gradients is an unbiased estimator for the expected gradient. Hence it is reasonable to expect that the norms of the large batch methods match to the ones of the small batch. And for a fair comparison, one should use the same learning rate for both training procedures. This suggests that a better comparison between GD and SGD (or LB and SB) should be scaled with the number of steps, so that, on average both algorithms are able to take similar number of steps of comparable sizes. Moreover, since random initial points in high dimensional spaces are almost always orthogonal, the fact that training with random initial points may lead to different basins should not be a surprising observation. However, the observation that LB converges to tighter basins that is separated by wells from the wider basins found by SB has been an observation that triggered attention. To test this idea even further we conduct the following experiment:

1. Part I: Train full CIFAR10 data for a bare AlexNet (no momentum, no dropout, no batch normalization) with a batch-size of 1,000. Record every 100 steps for 250 times.

---

3 Another important observation in this work that will be useful for our purposes is that constant rate GD (as well as SGD) do not stop as training progresses, the loss decreases ever so slightly. This means that even in the flat region, there is a small amount of signal coming from the total gradients that allow the algorithm to keep moving on the loss surface. We emphasize that this doesn’t contradict with the theory, but this implies that the convergence may be achieved at time scales much larger than the ones we observe in practice.
2. **Part II**: Continue training from the end point of the previous step with a smaller batch-size of 32. Everything else, including the constant learning rate is kept the same. And train another 250 periods each of which with 100 steps.

![Continuous training in two phases](image)

Figure 9: Large batch training immediately followed by small batch training on the full dataset of CIFAR10 with a raw version of AlexNet.

The key observation is the jump in the training and test losses, and a drop in the corresponding accuracies (Figure 9). Toward the end of **Phase II** the small batch reaches to a slightly better accuracy (about $\sim 1\%$). And this looks in line with the observations in Keskar et al. (2016), in that, it appears that the LB solution and SB solutions are separated by a barrier, and that the latter of which generalizes better. Moreover, the line interpolations extending away from either end points appear to be confirming the sharpness of LB solution. However, we find the straight line interpolation connecting the end points of **Part I** and **Part II** turns out to not contain any barriers (Figure 10). This suggests that while the small and large batch method converge to different solutions, these solutions have been in the same basin all along. This raises the striking possibility that that other seemingly different solutions may be similarly connected by a flat region to form a larger basin.

![Line interpolation between end points of the two phases](image)

Figure 10: Loss and accuracy evaluation on the straight line that contains LB and SB solutions. The accuracy of the SB solution is $\sim 1\%$ better than the LB solution, but there is no barrier between the two points.

One way to interpret the results goes through the Gauss-Newton decomposition introduced earlier. When we decrease the batch size, we increase the noise in the covariance of the gradients, and hence the first term starts to dominate. Even when the weight space has large flat regions, the fluctuations
of the stochastic noise should be precisely in the directions of the large eigenvalues. Therefore, at the beginning of Part II, the loss increases because the point fluctuates in the directions corresponding to the large eigenvalues, and eventually settles at a point that lies at the interior of the same level set, essentially staying in the same basin.

4.2 FURTHER DISCUSSIONS

One of the most striking implications of flatness may be the connected structure of the solution space. We may wonder whether two given solutions can be connected by a continuous path of solutions. This question have been explored in a recent work: in Freeman & Bruna (2016) it shown that for one hidden layer rectified neural networks the solution space is connected which is consistent with the flatness of the landscape.

The classical notion of basins of attractions may not be the suitable objects to study for neural networks. Rather, we may look at the exploration of interiors of level sets of the landscape. We may be tempted to speculate that such an exploration may indeed result in point that generalize better. However, the flat space itself is very high dimensional which comes with its own computational issues.

The training curve can be seen as composed of two parts: (1) high gain part where the norm of the gradients are large, (2) noise of the gradients is larger relative to the size of the stochastic gradients (see Shwartz-Ziv & Tishby (2017) for a recent reference). We speculate that the first part is relatively easy and even a large batch method can locate a large level set that contains points that generalize better than what’s initially found.

From a practical point of view, using larger batches with larger step sizes can in fact accelerate training. An example of this can be found in Goyal et al. (2017), where training Imagenet with a minibatch size of 8192 can match small batch performance.

On a final note for further consideration, we remark that we used standard pre-processing and initialization methods that are commonly used in practice. Fixing these two aspects, we modified the data, model, and algorithm in order to study their relative effects. However, the effects of pre-processing and initialization on the Hessian is highly non-trivial, and deserves a separate attention.

5 CONCLUSION

We have shown that the level of singularity of the Hessian cannot be ignored from theoretical considerations. Furthermore, we use the generalized Gauss-Newton decomposition of the Hessian to argue the cluster of zero eigenvalues are to be expected in practical applications. This allows us to reconsider the division between initial fast decay and final slow progress of training. We see that even large batch methods are able to get to the level where small batch methods go. As opposed to the common intuition, the observed generalization gap between the two is not due to small batch finding a different, better, wider basin. Instead, the two solutions appear to be in the same basin. This lack of barrier between solutions is demonstrated by finding paths between the two points that lie in the same level set. To conclude, we propose a major shift in perspective in considerations of the energy landscape in deep learning problems.

REFERENCES


A OUTLIER EIGENVALUES

In the subsequent experiments, we used a feed-forward neural network with a 100 dimensional input layer, two hidden layers each of which with 30 hidden units, and a \( k \)-dimensional output layer that is combined with softmax for \( k \)-class classification. We sampled random \( k \) Gaussian clusters in the input space and normalized the data globally. Then we carried out the training for the following sets of parameters: \( k : \{2, 5, 10, 20, 50\} \), algorithm: \{GD, SGD\}, non-linearity: \{tanh, ReLU\}, initial multiplier for the covariance of the input distribution: \{1, 10\}. Then we counted the number of large eigenvalues according to three different cutoff methods: largest consecutive gap, largest consecutive ratio, and a heuristic method of determining the threshold by searching for the elbow in the scree plot (see Figure 3). In Table 1 we marked the ones that are off by ±1.

<table>
<thead>
<tr>
<th>alg</th>
<th>size</th>
<th>cov</th>
<th>mean &amp; std</th>
<th>nc</th>
<th>gap</th>
<th>ratio</th>
<th>heur</th>
<th>cutoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>tanh</td>
<td>4022</td>
<td>1.0</td>
<td>1.19e-05, 1.19e-05</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>GD</td>
<td>tanh</td>
<td>4115</td>
<td>1.0</td>
<td>2.52e-05, 2.52e-05</td>
<td>5</td>
<td>2</td>
<td>17</td>
<td>N/A</td>
</tr>
<tr>
<td>GD</td>
<td>tanh</td>
<td>4270</td>
<td>1.0</td>
<td>3.75e-05, 3.75e-05</td>
<td>10</td>
<td>2</td>
<td>4</td>
<td>N/A</td>
</tr>
<tr>
<td>GD</td>
<td>tanh</td>
<td>4580</td>
<td>1.0</td>
<td>5.37e-05, 5.37e-05</td>
<td>20</td>
<td>3</td>
<td>4</td>
<td>N/A</td>
</tr>
<tr>
<td>GD</td>
<td>ReLU</td>
<td>4022</td>
<td>1.0</td>
<td>7.13e-06, 7.13e-06</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>GD</td>
<td>ReLU</td>
<td>4115</td>
<td>1.0</td>
<td>1.55e-05, 1.55e-05</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>GD</td>
<td>ReLU</td>
<td>4270</td>
<td>1.0</td>
<td>3.04e-05, 3.04e-05</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>GD</td>
<td>ReLU</td>
<td>4580</td>
<td>1.0</td>
<td>5.65e-05, 5.65e-05</td>
<td>20</td>
<td>4</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>GD</td>
<td>ReLU</td>
<td>5510</td>
<td>1.0</td>
<td>1.16e-04, 1.16e-04</td>
<td>50</td>
<td>50</td>
<td>51</td>
<td>49</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>4022</td>
<td>1.0</td>
<td>8.96e-06, 8.96e-06</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>4115</td>
<td>1.0</td>
<td>1.62e-05, 1.62e-05</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>4270</td>
<td>1.0</td>
<td>2.97e-05, 2.97e-05</td>
<td>10</td>
<td>2</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>4580</td>
<td>1.0</td>
<td>4.53e-05, 4.53e-05</td>
<td>20</td>
<td>2</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>5510</td>
<td>1.0</td>
<td>6.78e-05, 6.78e-05</td>
<td>50</td>
<td>50</td>
<td>51</td>
<td>49</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>4022</td>
<td>1.0</td>
<td>9.49e-05, 9.49e-05</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>4115</td>
<td>1.0</td>
<td>1.68e-04, 1.68e-04</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>4270</td>
<td>1.0</td>
<td>1.92e-04, 1.92e-04</td>
<td>10</td>
<td>2</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>4580</td>
<td>1.0</td>
<td>3.10e-04, 3.10e-04</td>
<td>20</td>
<td>20</td>
<td>21</td>
<td>19</td>
</tr>
<tr>
<td>SGD</td>
<td>ReLU</td>
<td>5510</td>
<td>1.0</td>
<td>1.67e-04, 1.67e-04</td>
<td>50</td>
<td>50</td>
<td>51</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 1: Counting outliers for matching the number of blobs. Dictionary of table elements: {cov: scale of covariance for inputs, mean&std: of the eigenvalues, nc: number of clusters, gap: largest consecutive gaps, ratio: largest consecutive ratios, heur: heuristic threshold, cutoff: the value of the heur.}

B THE SPECTRUM OF THE GENERALIZED GAUSS-NEWTON MATRIX

In this section, we will show that the spectrum of the Generalized Gauss-Newton matrix can be characterized theoretically under some conditions. Suppose that we can express the scaled gradient as \( g = Tx \) with the matrix \( T \in M \times d \) depending only on the parameters \( w \) - which is the case for linear models. Then we can write \( G \) as

\[
G = \frac{1}{N} \sum_{i \in D} g_i g_i^T = \frac{1}{N} TXX^T T^T
\]

where \( X = \{x^1, \ldots, x^N\} \) is an \( d \times N \) matrix Furthermore, without loss of generality, we assume that the examples are normalized such that the entries of \( X \) are independent with zero mean and unit variance. One of the first steps in studying \( G \) goes through understanding its principle components. In particular, we would like to understand how the eigenvalues and eigenvectors of \( G \) are related to the ones of \( \Sigma \) where \( \Sigma := \mathbb{E}(G) = \frac{1}{N} TXX^T T^T = TT^T \).

In the simplest case, we have \( \Sigma = I_d \) so that the gradients are uncorrelated and the eigenvalues of \( G \) are distributed according to the Marchenko-Pastur law in the limit where \( N, M \to \infty \) and \( \alpha := \frac{M}{N} \). The result dates back to sixties and can be found in (Marčenko & Pastur, 1967). Note that if \( M > N \) then there are \( M - N \) trivial eigenvalues of \( G \) at zero. Also, the width of the nontrivial distribution essentially depends on the ratio \( \alpha \). Clearly, setting the expected covariance to identity
is very limiting. One of the earliest relaxations appear in [Baik et al. 2005]. They prove a phase transition for the largest eigenvalues of the sample covariance matrix which has been known as the BBP phase transition. A case that may be useful for our setup is as follows:

**Theorem 1** [Baik, Arous, Péché, et al. 2005]. If $\Sigma = \text{diag}(\ell, 1, \ldots, 1)$, $\ell > 1$, and $M, N \to \infty$ with $\frac{M}{N} = \alpha \geq 1$. Let $c = 1 + \sqrt{\alpha}$, and let’s call the top eigenvalue of the sample covariance matrix as $\lambda_{\text{max}}$ then:

- If $1 \leq \ell < c$ then $\lambda_{\text{max}}$ is at the right edge of the spectrum with Tracy-Widom fluctuations.
- If $c < \ell$ then $\lambda_{\text{max}}$ is an outlier that is away from bulk centered at $\ell(1 + \frac{\alpha}{\ell^2})$ with Gaussian fluctuations.

Typically, due to the correlations in the problem we don’t have $\Sigma$ to be the identity matrix or a diagonal matrix with spikes. This makes the analysis of their spectrum a lot more difficult. A solution for this slightly more general case with non-trivial correlations has been provided only recently by [Bloemendal et al. 2016]. We will briefly review these results here see how they are related to the first term of the above decomposition.

**Theorem 2** [Bloemendal, Knowles, Yau, and Yin 2016]. If $d = M$, $\Sigma - I$ has bounded rank, $\log N$ is comparable to $\log M$, and entries of $X$ are independent with mean zero and variance one, then the spectrum of $\Sigma$ can be precisely mapped to the one of $G$ as $M, N \to \infty$ for fixed $\alpha = \frac{M}{N}$. Let $K = \min\{M, N\}$, and the decomposition of the spectrum can be described as follows:

- **Zeros**: $M - K$ many eigenvalues located at zero (if $M > N$).
- **Bulk**: Order $K$ many eigenvalues are distributed according to Marčenko-Pastur law.
- **Right outliers**: All eigenvalues of $\Sigma$ that exceed a certain value produce large-positive outlier eigenvalues to the right of the spectrum of $G$.
- **Left outliers**: All eigenvalues of $\Sigma$ that are close to zero produce small outlier eigenvalues between 0 and the left edge of the bulk of $G$.

Moreover, the eigenvectors of outliers of $G$ are close to the corresponding ones of $\Sigma$. 