The break-even point on the optimization trajectories of deep neural networks

Anonymous authors
Paper under double-blind review

Abstract

Understanding the optimization trajectory is critical to understand training of deep neural networks. We show how the hyperparameters of stochastic gradient descent influence the covariance of the gradients ($\mathbf{K}$) and the Hessian of the training loss ($\mathbf{H}$) along this trajectory. Based on a theoretical model, we predict that using a high learning rate or a small batch size in the early phase of training leads SGD to regions of the parameter space with (1) reduced spectral norm of $\mathbf{K}$, and (2) improved conditioning of $\mathbf{K}$ and $\mathbf{H}$. We show that the point on the trajectory after which these effects hold, which we refer to as *the break-even point*, is reached typically early during training. We demonstrate these effects empirically for a range of deep neural networks applied to multiple different tasks. Finally, we apply our analysis to networks with batch normalization (BN) layers and find that it is necessary to use a high learning rate to achieve loss smoothing effects attributed previously to BN alone.

1 Introduction

The choice of the optimization method implicitly regularizes deep neural networks (DNNs) by influencing the optimization trajectory in the loss surface (Neyshabur, 2017; Arora, 2019). In this work, we theoretically and empirically investigate how the learning rate and the batch size used at the beginning of training determine properties of the entire optimization trajectory.

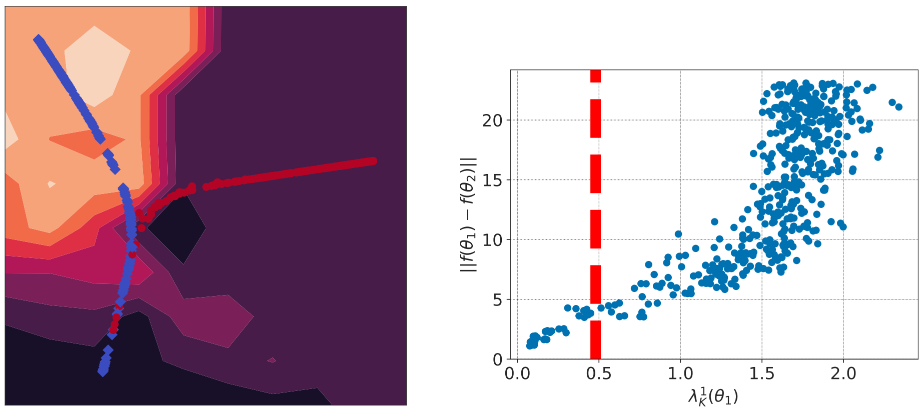


Figure 1: **Left**: Early part of the training trajectory on CIFAR-10 (before reaching 65% training accuracy) of a simple CNN model optimized using SGD with learning rate $\eta = 0.1$ (red) and $\eta = 0.01$ (blue). Each point (model) on the trajectory is represented by its test predictions embedded into a two-dimensional space using UMAP. The background color indicates the spectral norm of $\mathbf{K}$ (brighter is higher). Depending on η, after reaching what we call the break-even point, trajectories are steered towards regions characterized by different $\mathbf{K}$. **Right:** The spectral norm of $\mathbf{K}$ along the trajectory for $\eta = 0.01$ against the distances to the closest point from the trajectory for $\eta = 0.1$ (y axis). Vertical line marks the highest spectral norm of $\mathbf{K}$ achieved along the trajectory for $\eta = 0.1$.

We focus our analysis on two objects that quantify different properties of the optimization trajectory: the Hessian of the training loss ($\mathbf{H}$), and the covariance of gradients ($\mathbf{K}$).[1] The matrix $\mathbf{K}$ quantifies noise induced by noisy estimate of the full-batch gradient and has been also linked to generalization error (Roux et al., 2008; Fort et al., 2019). The matrix $\mathbf{H}$ describes the curvature of the loss surface and is often connected to optimization speed. Further, better conditioning of $\mathbf{H}$ and $\mathbf{K}$ has been attributed as the main reason behind the efficacy of batch normalization (Bjorck et al., 2018; Ghorbani et al., 2019).

Our first and main contribution is predicting and empirically demonstrating two effects induced in the early phase of training by the choice of the hyperparameters in stochastic gradient descent (SGD): (1) reduced spectral norms of $\mathbf{K}$ and $\mathbf{H}$ and (2) improved conditioning of $\mathbf{K}$ and $\mathbf{H}$. These effects manifest themselves after a certain point on the optimization trajectory, to which we refer to as *the break-even point*. See Fig. 1 for an illustration of this phenomenon. We make our predictions based on a theoretical model of the initial phase of training, which incorporates recent observations on the instability and oscillations in the parameter space that characterize the learning dynamics of neural networks (Masters & Luschi, 2018; Xing et al., 2018; Lan et al., 2019).

As our second contribution, we apply our analysis to a network with batch normalization (BN) layers and find that our predictions are valid in this case too. Delving deeper in this direction of investigation, we show that using a large learning rate is necessary to reach well-conditioned regions of the loss surface, which was previously attributed to BN alone (Bjorck et al., 2018; Ghorbani et al., 2019; Page, 2019).

2 Related work

Learning dynamics and the early phase of training. Our theoretical model is motivated by recent work on the learning dynamics of neural networks (Goodfellow et al., 2014; Masters & Luschi, 2018; Wu et al., 2018; Yao et al., 2018; Xing et al., 2018; Jastrzebski et al., 2018; Lan et al., 2019). We are most directly inspired by Xing et al. (2018); Jastrzebski et al. (2018); Lan et al. (2019) who show that training oscillates in the parameter space, and by Wu et al. (2018) who proposes a linear stability approach to studying how SGD selects a minimum.

In our work we argue that the initial phase of training has important implications for the rest of the trajectory. This is directly related to Erhan et al. (2010); Achille et al. (2017) who propose the existence of the critical period of learning. Erhan et al. (2010) argue that initial training, unless pre-training is used, is sensitive to shuffling of examples in the first epochs of training. Achille et al. (2017); Golatkar et al. (2019); Sagun et al. (2017); Keskar et al. (2017) demonstrate that adding regularizers in the beginning of training affects the final generalization disproportionately more compared to doing so later.

The covariance of the gradients and the Hessian. The covariance of the gradients, which we denote by $\mathbf{K}$, encapsulates the geometry and magnitude of variation in gradients across different samples (Thomas et al., 2019). The matrix $\mathbf{K}$ was related to the generalization error in Roux et al. (2008). A similar quantity, cosine alignment between gradients computed on individual examples, was recently shown to explain some aspects of deep networks generalization (Fort et al., 2019).

The second object that we study is the Hessian that quantifies the loss surface shape (LeCun et al., 2012). Recent work has shown that the largest eigenvalues of $\mathbf{H}$ grow quickly initially, and then stabilize at a value dependent on the learning rate and the batch size (Keskar et al., 2017; Sagun et al., 2017; Fort & Scherlis, 2019; Jastrzebski et al., 2018). The Hessian can be decomposed into a sum of two terms, where the dominant term (at least at the end of training) is the uncentered covariance of gradients $\mathbf{G}$ (Sagun et al., 2017; Papyan, 2019). While we study $\mathbf{K}$, the centered version of $\mathbf{G}$, $\mathbf{K}$ and $\mathbf{G}$ are typically similar due to the dominance of noise in training (Zhu et al., 2018; Thomas et al., 2019).

Implicit regularization induced by optimization method. Multiple prior work study the regularization effects that are attributed only to the optimization method (Neyshabur, 2017). A popular

[1]We define it as $\mathbf{K} = \frac{1}{N}\sum_{i=1}^{N}(g_i - g)^T(g_i - g)$, where $g_i = g(\mathbf{x_i}, y_i; \theta)$ is the gradient of $\mathcal{L}$ with respect to θ calculated on i-th example, N is the number of training examples, and g is the full-batch gradient.

research direction is to bound the generalization error based on the properties of the final minimum such as the norm of the parameter vector or the Hessian (Bartlett et al., 2017; Keskar et al., 2017). Perhaps the most related work to ours is (Arora et al., 2019; Arora, 2019). They suggest it is necessary to study the trajectory to understand generalization of deep networks. In this vein, but in contrast to most of the previous work, we focus (1) on the implicit regularization effects that can be attributed to the GD dynamics at the beginning of training, and (2) on the covariance of gradients.

3 Two conjectures about SGD trajectory

In this section we make two conjectures about the optimization trajectory induced by SGD based on a theoretical model of the learning dynamics in the early stage of training.

Definitions. Let us denote loss on an example $(\mathbf{x}, y)$ by $\mathcal{L}(\mathbf{x}, y; \theta)$, where θ is a D-dimensional parameter vector. A key object we study is the Hessian $\mathbf{H}$ of the training loss. The second key object we study is the covariance of the gradients $\mathbf{K} = \frac{1}{N}\sum_{i=1}^{N}(g_i - g)^T(g_i - g)$, where $g_i = g(\mathbf{x_i}, y_i; \theta)$ is the gradient of $\mathcal{L}$ with respect to θ calculated on i-th example, N is the number of training examples, and g is the full-batch gradient. We denote the i-th normalized eigenvector and eigenvalue of a matrix $\mathbf{A}$ by e_A^i and λ_A^i. Both $\mathbf{H}$ and $\mathbf{K}$ are computed at a given θ, but we omit this dependence in the notation. Let t index steps of optimization, and $\theta(t)$ the parameter vector at optimization step t.

Inspired by Wu et al. (2018) we study stability of optimization, in our case restricted to e_H^1. Let us call the projection of parameters θ onto e_H^1 by $\psi = \langle \theta, e_H^1 \rangle$. With a slight abuse of notation let $g(\psi) = \langle g(\theta), e_H^1 \rangle$. Similarly to Wu et al. (2018), we say SGD is *unstable along* e_H^1 at $\theta(t)$ if the norm of elements of sequence $\psi(\tau + 1) = \psi(\tau) - \eta g(\psi(\tau))$ diverges when $\tau \to \infty$, where $\psi(0) = \theta(t)$. The sequence $\psi(\tau)$ represents optimization trajectory restricted from $\theta(t)$ only to the direction of e_H^1.

Assumptions. We make the following assumptions to build our model:

1. The loss surface for each example projected onto e_H^1 is a quadratic function. This assumption is also used by Wu et al. (2018). It was shown to hold for the loss averaged over examples by Alain et al. (2019). It is also well known that the spectral norm of $\mathbf{H}$ is positive (Sagun et al., 2017).
2. The eigenvectors e_H^1 and e_K^1 are co-linear, i.e. $e_H^1 = \pm e_K^1$, and furthermore $\lambda_K^1 = \alpha \lambda_H^1$ for some $\alpha \in \mathbb{R}$. This is inspired by Papyan (2019) who show that $\mathbf{H}$ can be approximated by $\mathbf{G}$ (uncentered $\mathbf{K}$).
3. When in a region that is not stable along e_H^1, training trajectory steers towards more stable regions by decreasing λ_H^1. This is inspired by recent work showing training can escape region with too large curvature compared to the learning rate (Zhu et al., 2018; Wu et al., 2018; Jastrzebski et al., 2018).
4. The spectral norm of $\mathbf{H}$, λ_1^H, increases during training, unless increasing λ_1^H would lead to entering a region where training is not stable along e_H^1. This is inspired by (Keskar et al., 2017; Sagun et al., 2017; Jastrzebski et al., 2018; Fort & Scherlis, 2019) who show that in many settings λ_1^H increases in the beginning of training.

These assumptions are only used to build a theoretical model for the early phase of training. Its main purpose is to make predictions about the training procedure that we verify empirically.

Reaching the break-even point earlier for a larger learning rate or a smaller batch size. Let us restrict to the case when training is initialized at $\theta(0)$ at which SGD is stable along $e_H^1(0)$.[2] We aim to show that the learning rate (η) and the batch size (S) determine $\mathbf{H}$ and $\mathbf{K}$ in our model, and conjecture that the same holds in real neural networks.

Consider two optimization trajectories for η_1 and η_2, where $\eta_1 > \eta_2$, that are initialized at the same θ_0, where optimization is stable along $e_H^1(t)$ and $\lambda_H^1(t) > 0$. Under Assumption 1 the loss surface

[2]We include in App. A a similar argument for the opposite case.

along $e_H^1(t)$ can be expressed as $f(\psi) = \sum_{i=1}^{N} (\psi - \psi^*)^2 H_i(t)$, where $H_i(t) \in \mathbb{R}$. It can be shown that at any iteration t the necessary and sufficient condition for SGD to be stable along $e_H^1(t)$ is:

$$(1 - \eta\lambda_1^H(t))^2 + s(t)^2 \frac{\eta^2(N - S)}{S(N - 1)} \leq 1, \quad (1)$$

where N is the training set size and $s(t)^2 = \mathrm{Var}[H_i(t)]$ over the training examples. A proof can be found in (Wu et al., 2018). We call this point on the trajectory on which the LHS of Eq. 1 becomes equal to 1 for the first time the *break-even point*.

Under the Assumption 3, $\lambda_1^H(t)$ and $\lambda_1^K(t)$ increase over time. If $s = N$, the break-even point is reached at $\lambda_1^H(t) = \frac{2}{\eta}$. More generally, it can be shown that for η_1, the break-even point is reached for a lower magnitude of $\lambda_1^H(t)$ than for η_2. The same reasoning can be carried out for S. We state this formally and prove in App. A.

From this point on the trajectory, under Assumption 4, SGD does not enter regions where either $\lambda_1^H(t')$ or $\lambda_1^K(t')$ is larger than at the break-even point, as otherwise it would lead to increasing one of the terms in LHS of Eq. 1, and hence losing stability along $e_H^1(t')$.

The two conjectures about real DNNs. Assuming that real DNNs reach the break-even point, we make the following two conjectures, arising from our theoretical model above, about their optimization trajectory. The most direct implication of reaching the break-even point is that λ_K^1 and λ_H^1 at the break-even point depend on η and S, which we formalize as:

Conjecture 1 (Variance reduction effect of SGD)**.** *Along the SGD trajectory, the maximum attained values of λ_H^1 and λ_K^1 are smaller for a larger learning rate or a smaller batch size.*

We refer to Con. 1 as *variance reduction effect of SGD*, because reducing λ_K^1 can be shown to reduce the L_2 distance between the full-batch gradient, and the mini-batch gradient.

Next, we make another, stronger, conjecture. It is plausible to assume that reaching the break-even point does not affect the λ_H^i and λ_K^i for $i \neq 1$, because increasing their values does not impact stability along e_H^1 in our theoretical model. It is also well known that a large number of eigenvalues of $\mathbf{H}$ increase initially (Fort & Scherlis, 2019; Sagun et al., 2017). Based on these remarks we conjecture that:

Conjecture 2 (Pre-conditioning effect of SGD)**.** *Along the SGD trajectory, the maximum attained values of $\frac{\lambda_K^*}{\lambda_K^1}$ and $\frac{\lambda_H^*}{\lambda_H^1}$ are larger for a larger learning rate or a smaller batch size, where λ_K^* and λ_H^* are the smallest nonzero eigenvalues of $\mathbf{H}$ and $\mathbf{K}$, respectively. Furthermore, the maximum attained values of $\mathrm{Tr}(\mathbf{K})$, $\mathrm{Tr}(\mathbf{H})$ are smaller for a larger learning rate or a smaller batch size.*

We consider non-zero eigenvalues in the conjecture, because K has $N - D$ non-zero eigenvalues, where N is the number of training points. In practice we will measure $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$, which is an upper-bound on λ_K^*/λ_K^1.

4 EXPERIMENTS

In this section we first analyse learning dynamics in the early phase of training. Next, we empirically validate the two conjectures. In the final part we extend our analysis to a neural network with batch normalization layers.

Due to the space constraint we take the following approach to reporting results. In the main body of the paper, we focus on the CIFAR-10 dataset (Krizhevsky, 2009) and the IMDB dataset (Maas et al., 2011), to which we apply three architectures: a vanilla CNN (SimpleCNN) following Keras example (Chollet et al., 2015), ResNet-32 (He et al., 2015a), and LSTM (Hochreiter & Schmidhuber, 1997). We also validate the two conjectures for DenseNet (Huang et al., 2016) on the ImageNet (Deng et al., 2009) dataset, BERT (Devlin et al., 2018b) fine-tuned on the MNLI dataset (Williams et al., 2017), and a multi-layer perceptron on the FashionMNIST dataset (Xiao et al., 2017). These results are in the Appendix. We include all experimental details in App. C.

Following Dauphin et al. (2014); Alain et al. (2019), we estimate the top eigenvalues and eigenvectors of $\mathbf{H}$ using the Lanczos algorithm on a random subset of 5% of the training set on CIFAR-10.

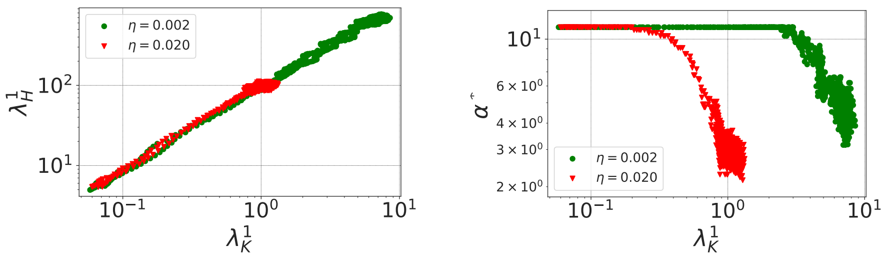


Figure 2: The evolution of λ_K^1 (spectral norm of $\mathbf{K}$), λ_H^1 (spectral norm of $\mathbf{H}$), and α^* (width of the loss surface, see text for details) in the early phase of training. Consistently with our theoretical model, λ_K^1 is correlated initially with λ_H^1 (left) and α^1 (right). The training reaches a smaller maximum value of λ_K^1 and λ_H^1 for a higher learning rate.

We estimate the top eigenvalues and eigenvectors of $\mathbf{K}$ using (in most cases) batch size of 128 and approximately 5% of the training set on CIFAR-10. We describe the procedure in more details, as well as compare to using batch size of 1, in App. B.

4.1 A CLOSER LOOK AT THE EARLY PHASE OF TRAINING

In this section we examine the learning dynamics in the early phase of training. Our goal is to verify some of the assumptions made in Sec. 3. We analyse the evolution of λ_H^1 and λ_K^1 for $\eta = 0.02$ and $\eta = 0.2$ using the SimpleCNN on the CIFAR-10 dataset.

Are λ_K^1 and λ_H^1 correlated in the beginning of training? The key assumption behind our theoretical model is that λ_K^1 and λ_H^1 are correlated, at least prior to reaching the break-even point. We confirm this in Fig. 2. The highest achieved λ_K^1 and λ_H^1 are larger for the smaller η. Additionally, we observe that after achieving the highest value of λ_H^1, further growth of λ_K^1 does not translate into an increase of λ_K^1. This is expected: λ_H^1 decays to 0 when the mean loss decays to 0 for cross entropy loss (Martens, 2016).

Does training become increasingly unstable in the early phase of training? A key aspect of our model is that an increase of λ_K^1 and λ_H^1 translates into a decrease in stability, which we formalized as stability along e_H^1. Computing stability directly along e_H^1 is computationally expensive. Instead, we measure a more tractable proxy. Let us define α^* to be the minimal increment of the SGD step length such that the training loss increases by at least 20%. More precisely, let $\theta(t)$ and $\theta(t+1)$ denote the two consecutive steps on the optimization trajectory. We define α^* as the minimum α such that $\mathcal{L}(\theta(t) - \alpha^*(\theta(t) - \theta(t+1))) \geq 1.2\mathcal{L}(\theta(t))$. In Fig. 2 we observe that α^* is anti-correlated with λ_K^1, which is consistent with our theoretical model. We also observe that α^* reaches a value close to 2 for both tested η, which suggests reaching the break-even point.

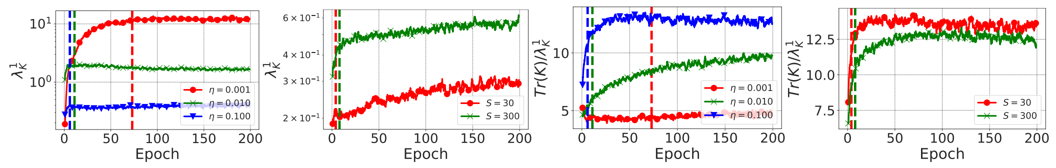
Visualizing the break-even point. Finally, to understand the break-even point phenomenon better, we visualize the learning dynamics leading to reaching the break-even point in our model in Fig. 1 (left). Following Erhan et al. (2010), we embed the test set predictions at each step of training of SimpleCNN, in our case using UMAP (McInnes et al., 2018). In the Figure we observe that the trajectory corresponding to $\eta = 0.01$ diverges from the trajectory corresponding to $\eta = 0.1$ when entering a region of the loss surface characterized by a high λ_K^1. Because a low dimensional embedding can mischaracterize the true distances (Wattenberg et al., 2016), we confirm our interpretation by plotting the L_2 distance to the closest iteration of $\eta = 0.1$ trajectory in the right panel of Fig. 1.

Summary. We have shown that the dynamics of the early phase of training is consistent with the assumptions made in our model. That is, λ_K^1 and λ_H^1 increase approximately proportionally to each other, which is also correlated with a decrease of a proxy of stability. Finally, we have shown qualitatively reaching the break-even point.

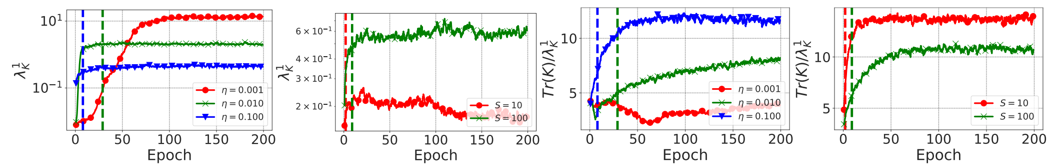
4.2 Variance reduction and pre-conditioning effect of SGD

In this section we validate empirically Con. 1 and Con. 2 in three settings. For each model we pick manually a suitable range of learning rates and batch sizes to ensure that the properties of $\mathbf{K}$ and $\mathbf{H}$ that we examine have converged in a reasonable computational budget; we use 200 epochs on CIFAR-10 and 50 epochs on IMDB.

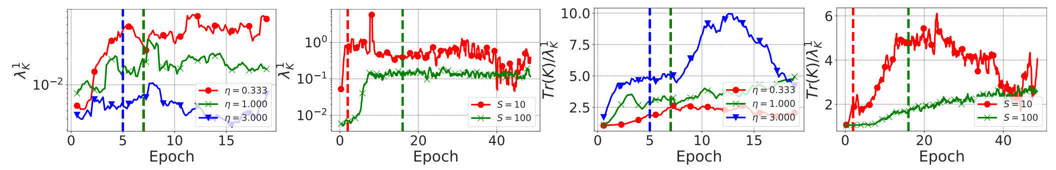
We summarize the results in Fig. 3, Fig. 4 for SimpleCNN, ResNet-32, and LSTM. Curves are smoothed with a moving average. The training curves, as well as experiments for other architectures and datasets (including a DenseNet on ImageNet and BERT on MNLI) can be found in App. D.



(a) SimpleCNN on the CIFAR-10 dataset



(b) ResNet-32 on the CIFAR-10 dataset



(c) LSTM on the IMDB dataset

Figure 3: The variance reduction and the pre-conditioning effect of SGD in various settings. Trajectories corresponding to higher learning rates (η) or lower batch sizes (S) are characterized by lower maximum λ_K^1 (variance reduction) and larger maximum $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$ (better conditioning). These effects occur early in training. Vertical lines mark the first epoch at which training accuracy is above 60% for CIFAR-10, and above for 75% for IMDB.

Null hypothesis. A natural assumption is that the choice of η or S does not influence $\mathbf{K}$ and $\mathbf{H}$ along the optimization trajectory. In particular, it is not self-evident that using a high η, or a small S, would steer optimization towards better conditioned regions of the loss surface.

Conjecture 1. To validate Conjecture 1 we examine the highest value of λ_K^1 observed along the optimization trajectory. As visible in Fig. 3 using a higher η results in λ_K^1 achieving a lower maximum. For instance $\max(\lambda_K^1) = 0.87$ and $\max(\lambda_K^1) = 3.01$ for $\eta = 0.1$ and $\eta = 0.01$, respectively. Similarly, we can conclude that using a higher S in SGD leads to reaching a higher value of λ_K^1.

Recall that we compute λ_K^1 using a constant batch size of 128. While we say that low S leads to variance reduction (lower maximum λ_K^1), this is not contradictory to the fact that increasing S generally decreases variance of mini-batch gradients.

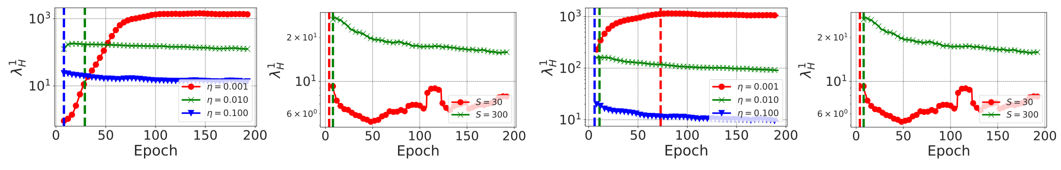
Conjecture 2. To test Conjecture 2 we compute the maximum value of $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$ along the optimization trajectory. It is visible in Fig. 3 that using a higher η results in a lower minimum value of $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$. For instance, $\max(\mathrm{Tr}(\mathbf{K})/\lambda_K^1) = 14.29$ and $\max(\mathrm{Tr}(\mathbf{K})/\lambda_K^1) = 10.69$ for $\eta = 0.1$ and $\eta = 0.01$, respectively. Similarly, we can conclude from these plots that using a higher S leads to lower $\max(\mathrm{Tr}(\mathbf{K})/\lambda_K^1)$.

Due to space constraints we move Figures showing the effect of η and S on $\mathrm{Tr}(\mathbf{K})$ to App. 3. We observe that the maximum of $\mathrm{Tr}(\mathbf{K})$ depends on η and S in the same way as λ_K^1.

How early in training is the break-even point reached? How λ_H^1 and λ_K^1 depend on η and S at the end of training was already studied by Jastrzebski et al. (2017); Keskar et al. (2017); Wu et al. (2018). Importantly, we find that λ_K^1 and λ_H^1 reach the highest value early in training: close to reaching 60% training accuracy on CIFAR-10, and 75% training accuracy on IMDB. See also the vertical lines in Fig. 3.

Other experiments. We report how λ_H^1 depend on η and S for ResNet-32 and SimpleCNN in Fig. 4. We observe that the conclusions carry over to λ_H^1, which is consistent with experiments in Jastrzebski et al. (2018). We found the effect on $\mathrm{Tr}(\mathbf{H})/\lambda_H^1$ of η and S to be weaker. This might be because, in contrast to $\mathrm{Tr}(\mathbf{K})$, we approximate $\mathrm{Tr}(\mathbf{H})$ using only the top five eigenvalues (see App. B for details).

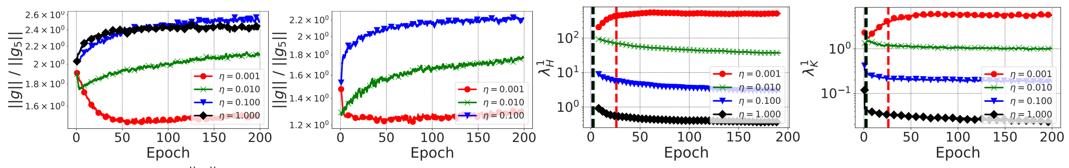
Summary In this section we have demonstrated the variance reduction (Conjecture 1) and the pre-conditioning effect (Conjecture 2) of SGD. Furthermore, we have shown these effects occur early in training. We also found that conclusions carry over to other settings including BERT on MNLI and DenseNet on ImageNet (see App. D).



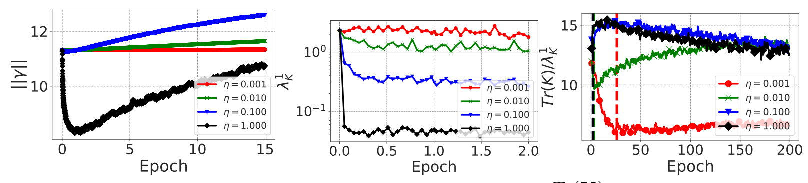
(a) ResNet-32 (b) SimpleCNN

Figure 4: The variance reduction of SGD, for ResNet-32 (left) and SimpleCNN (right). Trajectories corresponding to higher learning rates (η) or smaller batch size (S) are characterized by a lower maximum λ_H^1. Vertical lines mark the first epoch at which training accuracy is above 60%.

4.3 IMPORTANCE OF η FOR CONDITIONING IN BATCH NORMALIZED NETWORKS



(a) **Left**: $\frac{\|g\|}{\|g_5\|}$ for SimpleCNN-BN and SimpleCNN. **Right**: λ_H^1 and λ_K^1 for SimpleCNN-BN



(b) **Left**: $\|\gamma\|$ of the last BN layer. **Middle**: λ_K. **Right**: $\frac{\mathrm{Tr}(\mathbf{K})}{\lambda_K^1}$ for SimpleCNN-BN

Figure 5: Evolution of various metrics that quantify conditioning of the loss surface for SimpleCNN with and without batch normalization layers (SimpleCNN and SimpleCNN-BN), and for different learning rates.

The loss surface of deep networks has been widely reported to be ill-conditioned, which is the key motivation behind using second order methods in deep learning (LeCun et al., 2012; Martens &

Grosse, 2015). Recently, Ghorbani et al. (2019); Page (2019) have argued that the key reason behind the efficacy of batch normalization (Ioffe & Szegedy, 2015) is improving conditioning of the loss surface. Our Conjecture 2 is that using a high η or a low S results as well in improved conditioning. A natural question that we investigate in this section is how the two phenomena are related.

Are the two conjectures valid in batch normalized networks? First, to investigate whether our conjectures hold in batch normalized network, we run similar experiments as in Sec. 4.2 on a SimpleCNN model with batch normalization layers inserted after each convolutional layer (SimpleCNN-BN), using the CIFAR-10 dataset. We test $\eta \in \{0.001, 0.01, 0.1, 1.0\}$; $\eta = 1.0$ leads to divergence of SimpleCNN without BN. We summarize the results in Fig. 5. The evolution of $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$ and λ_K^1 show that *both Conjecture 1 and Conjecture 2 hold in this setting.*

A closer look at the early phase of training. To further corroborate that our analysis applies to BN networks, we study the early phase of training of a network with batch normalization layers, complementing the results in Sec. 4.1.

We observe in Fig. 5 (bottom) that training of SimpleCNN-BN starts in a region characterized by relatively high λ_K^1. This is consistent with prior work showing that batch normalized networks lead to gradient explosion in the first iteration (Yang et al., 2019). λ_K^1 then decays for all but the lowest η. This behavior is consistent with our theoretical model. We also track the norm of the scaling factor in BN, $\|\gamma\|$, in the last layer of the network in Fig. 5 (bottom). It is visible that $\eta = 1.0$ and $\eta = 0.1$ initially decrease the value of $\|\gamma\|$, which we hypothesize to be one of the mechanisms by which high η steers optimization towards better conditioned regions of the loss surface in BN networks.

BN requires using a high learning rate. As our conjectures hold for BN network, a natural question is if learning can be ill-conditioned with a low learning rate even when BN is used. Ghorbani et al. (2019) show that without BN, mini-batch gradients are largely contained in the subspace spanned by the top eigenvectors of noncentered $\mathbf{K}$. To answer this question we track $\|g\|/\|g_5\|$, where g denotes the mini-batch gradient, and g_5 denotes the mini-batch gradient projected onto the top 5 eigenvectors of $\mathbf{K}$. A value of $\|g\|/\|g_5\|$ close to 1 implies that the mini-batch gradient is mostly contained in the subspace spanned by the top 5 eigenvectors of $\mathbf{K}$.

We compare two settings: (1) SimpleCNN-BN optimized with $\eta = 0.001$, and (2) SimpleCNN optimized with $\eta = 0.01$. We make three observations. First, the maximum (minimum) value of $\|g\|/\|g_5\|$ is 1.90 (1.37) and 1.88 (1.12), respectively. Second, the maximum value of λ_K^1 is 10.3 and 16, respectively. Finally, $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$ reaches 12.14 in the first setting, and 11.55 in the second setting. Comparing these differences to differences that are induced by using the highest $\eta = 1.0$ in SimpleCNN-BN, we can conclude that using a large learning rate is necessary to observe the effect of loss smoothing which was previously attributed to BN alone (Ghorbani et al., 2019; Page, 2019; Bjorck et al., 2018). This might be directly related to the result that a high learning rate is necessary to achieve good generalization when using BN (Bjorck et al., 2018).

Summary. We have shown that our analysis applies to a network with batch normalization layers, and that *using a high learning rate is necessary in a batch normalized network to improve conditioning of the loss surface relatively to the same network without batch normalization.*

5 Conclusion

Based on a theoretical model, we conjectured and empirically argued for the existence of the break-even point on the optimization trajectory induced by SGD. Next, we demonstrated that using a high learning rate or a small batch size in SGD has two effects on $\mathbf{K}$ and $\mathbf{H}$ along the trajectory that we referred to as (1) variance reduction and (2) pre-conditioning.

There are many potential implications of the existence of the break-even point. We investigated one in particular, and demonstrated that using a high learning rate is necessary to achieve the loss smoothing effects previously attributed to batch normalization alone.

Additionally, the break-even occurs typically early during training, which might be related to the recently discovered phenomenon of the critical learning period in training of deep networks (Achille et al., 2017; Golatkar et al., 2019). We plan to investigate this connection in the future.

References

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep neural networks. *CoRR*, abs/1711.08856, 2017.

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

Sanjeev Arora. Is optimization a sufficient language for understanding deep learning? 2019.

Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. In *International Conference on Learning Representations*, 2019.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017.

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018.

François Chollet et al. Keras, 2015.

Yann N. Dauphin, Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *CoRR*, abs/1406.2572, 2014.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018a.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018b.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11, March 2010. ISSN 1532-4435.

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, 2019.

Stanislav Fort, Pawe Krzysztof Nowak, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks, 2019.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Time Matters in Regularizing Deep Networks: Weight Decay and Data Augmentation Affect Early Learning Dynamics, Matter Little Near Convergence. *arXiv e-prints*, art. arXiv:1905.13277, May 2019.

Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. Qualitatively characterizing neural network optimization problems. *arXiv e-prints*, art. arXiv:1412.6544, Dec 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015b.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8), November 1997. ISSN 0899-7667.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15. JMLR.org, 2015.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos J. Storkey. Three factors influencing minima in SGD. *CoRR*, abs/1711.04623, 2017.

Stanislaw Jastrzebski, 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 e-prints*, art. arXiv:1807.05031, Jul 2018.

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. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Janice Lan, Rosanne Liu, Hattie Zhou, and Jason Yosinski. LCA: Loss Change Allocation for Neural Network Training. *arXiv e-prints*, art. arXiv:1909.01440, Sep 2019.

Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient BackProp*, pp. 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_3. URL `https://doi.org/10.1007/978-3-642-35289-8_3`.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

James Martens. *Second-order optimization for neural networks*. University of Toronto (Canada), 2016.

James Martens and Roger B. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. *CoRR*, abs/1503.05671, 2015.

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29), 2018.

Behnam Neyshabur. Implicit regularization in deep learning. *CoRR*, abs/1709.01953, 2017.

David Page. How to train your resnet 7: Batch norm. 2019.

Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *CoRR*, abs/1901.08244, 2019.

Nicolas L. Roux, Pierre antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (eds.), *Advances in Neural Information Processing Systems 20*. Curran Associates, Inc., 2008.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015.

Levent Sagun, Utku Evci, V. Ugur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *CoRR*, abs/1706.04454, 2017.

Valentin Thomas, Fabian Pedregosa, Bart van Merriënboer, Pierre-Antoine Manzagol, Yoshua Bengio, and Nicolas Le Roux. Information matrices and generalization. *CoRR*, abs/1906.07774, 2019.

Martin Wattenberg, Fernanda Vigas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426, 2017.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018.

Lei Wu, Chao Ma, and Weinan E. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, USA, 2018. Curran Associates Inc.

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

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A Walk with SGD. *arXiv e-prints*, art. arXiv:1802.08770, Feb 2018.

Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. *CoRR*, abs/1902.08129, 2019.

Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. *CoRR*, abs/1802.08241, 2018.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Minima and Regularization Effects. *arXiv e-prints*, art. arXiv:1803.00195, Feb 2018.

A PROOFS

In this section we will formally state and prove the theorem used informally in Sec. 3. With the definitions introduced in Sec. 3 in mind, we propose the following:

Theorem 1. *Assuming that training is stable along $e_H^1(t)$ at $t = 0$, then $\lambda_1^H(t)$ and $\lambda_K^1(t)$ at which SGD becomes unstable along $e_H^1(t)$ are smaller for a larger η or a smaller S.*

Proof. This theorem is almost a direct implication of Theorem 1 from Wu et al. (2018) adapted to our context.

First, let us consider two experiments that follow the same trajectory with η_1 and η_2 ($\eta_1 > \eta_2$). More precisely, treating ψ as a function of λ_H^1, we assume $\psi(\lambda_H^1)$ are the same up to $\lambda_H^1(t^*)$ when training for one of them becomes unstable along $e_H^1(t^*)$. We also assume that $\lambda_K^1(t) = \alpha\lambda_H^1(t)$ for some $\alpha \in \mathbb{R}$.

Consider iteration t. Theorem 1 of Wu et al. (2018) states that SGD is stable at this iteration if the following is true:

$$(1 - \eta\lambda_1^H(t))^2 + s^2(t)\frac{\eta^2(n-S)}{S(n-1)} \leq 1, \tag{2}$$

where $s(t) = \mathrm{Var[H_i(t)]}$.

Recall that we consider a single dimensional quadratic loss surface $\mathcal{L}(\psi) = \frac{1}{2N}\sum_{i=1}^{N}(\psi_i - \psi^*)^2 H_i$, where $H_i, \psi^* \in \mathbb{R}$. Without loss of generality we assume $\psi^* = 0$.

Using the assumption that the loss is quadratic for each example we get $\mathrm{Var}[g_i(\psi)] = \mathrm{Var}[\psi H_i] = \psi^2 s(t)^2$. Hence, SGD is stable at given t along $e_H^1(t)$ if and only if:

$$(1 - \eta\lambda_1^H(t))^2 + \frac{\lambda_K^1(t)}{\psi(t)^2}\frac{\eta^2(n-S)}{S(n-1)} \leq 1. \tag{3}$$

First, let us assume without loss of generality that $\psi(t) = 1$ for all t. In this case, we can easily solve Eq. 3 for $\lambda_H^1(t^*)$. A simple algebraic manipulation leads to:

$$\lambda_H^1(t^*) = \frac{2 - \alpha\frac{(n-S)}{(n-1)}\frac{\eta}{S}}{\eta}. \tag{4}$$

Note that if $n = S$ the right hand side degenerates to $\frac{2}{\eta}$. We can conclude that for a larger S or smaller η, $\lambda_H^1(t^*)$ is lower.

□

It is also straightforward to extend the argument to the case when training is initialized at an unstable region along $e_H^1(0)$:

Theorem 2. *If training is unstable along $e_H^1(t)$ at $t = 0$, then $\lambda_1^H(t)$ and $\lambda_K^1(t)$ at which SGD becomes first stable along $e_H^1(t)$ are smaller for a larger η or a smaller S.*

Proof. In this case, by Assumption 4, training steers to a region where training is stable along e_H^0, which must result in reducing λ_H^1 and λ_K^1.

The value λ_H^1 at which training becomes stable is again $\lambda_H^1(t^*) = \frac{2-\alpha\frac{(n-S)}{(n-1)}\frac{\eta}{S}}{\eta}$. Based on this we conclude, that the value λ_H^1 and λ_K^1 at which training becomes stable depends on η and S.

□

B APPROXIMATING THE EIGENSPACE OF $\mathbf{K}$ AND $\mathbf{H}$

Due to the ill-conditioning of $\mathbf{H}$ a small subset of the training set approximates well the largest eigenvalues of the true Hessian on the CIFAR-10 dataset by (Alain et al., 2019), in their case 5% of the dataset. We use the same fraction in CIFAR-10 experiments. Following Alain et al. (2019) we use SCIPY Python package.

Evaluating the eigenspace of $\mathbf{K}$ is perhaps less common in deep learning. We sample L mini-batch gradient of size M. Then following Papyan (2019) we compute the corresponding Gram matrix $\hat{\mathbf{K}}$ that has entries $\hat{\mathbf{K}}_{ij} = \langle g_i - g, g_j - g \rangle$, where g is the full batch gradient. Note that $\hat{\mathbf{K}}_{ij}$ is only $L \times L$ dimensional. It can be shown that in expectation of mini-batches $\hat{\mathbf{K}}$ has the same eigenspectrum as $\mathbf{K}$. It can be shown that $Tr(\hat{\mathbf{K}}) = Tr(\mathbf{K})$. We compute the full spectrum using SVD procedure from SCIPY Python package.

In all experiments we fix $L = 25$. When comparing different batch sizes we use $M = 128$, otherwise we use the same batch size as the one used to train the model. For instance on the CIFAR-10 dataset this amounts to using approximately 5% of the training set.

To compute $\mathrm{Tr}(\mathbf{K})$ we compute the trace of $\hat{\mathbf{K}}$. Due to large computational cost of estimating top eigenvalues using the Lanczos algorithm, we approximate the $Tr(\mathbf{H}$ using the top 5 eigenvalues of $\mathbf{H}$

A natural question is whether using $M = 128$ approximates well the underlying $\mathbf{K}$. To investigate this we compare λ_K^1 evaluated using either $M = 128$ or $M = 1$. We adapt L to keep the same number of $N = LM$ of used training examples in the computation. We test this on SimpleCNN model with $\eta = 0.01$. We show that the two are strongly correlated in Fig. 6, which we conclude to be enough for our analysis.

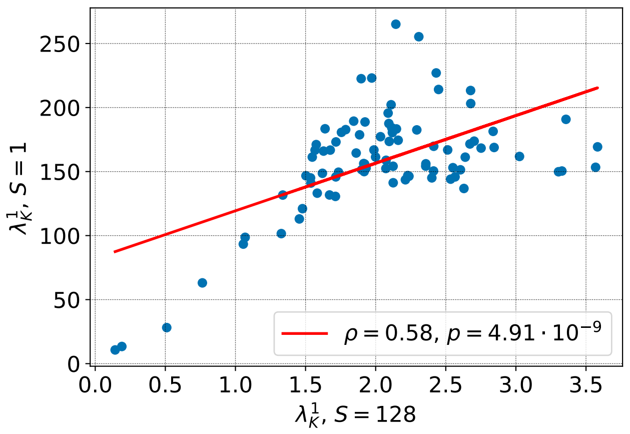


Figure 6: Correlation between values of $\mathbf{K}$ obtained using different mini-batch sizes (SimpleCNN model on CIFAR-10).

C EXPERIMENTAL DETAILS

In this section we describe all the details for experiments in the main text and in the Appendix.

ResNet-32 on CIFAR The model trained is the **ResNet-32** (He et al., 2015b). The network is trained for 200 epochs with a batch size equal to 128. The dataset used is CIFAR-10. Standard data augmentation and preprocessing is applied. The default learning rate is 0.05, and the default batch size is 128. Weight decay 0.0001 is applied to all convolutional layers.

SimpleCNN on CIFAR The network used is a simple convolutional network based on example from Keras repository (Chollet et al., 2015). First, the data is passed through two convolutional layers with 32 filters, both using *same* padding, ReLU activations, and the kernel size of 3x3. Next, the second pair of layers is used with 64 filters (the rest of parameters are the same). Each pair ends with a max-pooling layer with a window of size 2. Before the classification layer, a densely connected layer with 128 units is used. The data follows the same scheme as in the ResNet-32 experiment. The default learning rate is 0.05, and the default batch size is 128.

BERT on MNLI The model used in this experiment is **BERT-base** (Devlin et al., 2018a), pre-trained on multilingual data[3]. The model is trained on MultiNLI dataset (Williams et al., 2018) with the maximum sentence length equal to 40. The network is trained for 20 epochs with a batch size of 32.

MLP on FMNIST This experiment is using a Multi Layer Perceptron network with two hidden layers of size 300 and 100, both with ReLU activations. The data is normalized to the $[0, 1]$ range and no augmentation is being used. The network is trained with a batch size of 64 for 200 epochs.

LSTM on IMDB The network used in this experiment consists of an embedding layer followed by an LSTM with 100 hidden units. We use vocabulary size of 20000 words and the maximum length of the sequence equal to 80. The model is trained for 100 epochs with a batch size of 128.

DenseNet on ImageNet The network used is the **DenseNet-121** (Huang et al., 2016). The dataset used is the ILSVRC 2012 (Russakovsky et al., 2015). The images are centered, but no augmentation is being used. Due to large computational cost, the network is trained for 10 epochs with a batch size of 32. Neither dropout nor weight decay is used for training.

D Additional experiments for Sec. 4.2

In this section we repeat experiments from Sec. 4.2 in various other settings, as well as include additional data from settings already included in the main text.

ResNet-32 on CIFAR-10. In Fig. 7 and Fig. 8 we report accuracy on the training set and the validation set, λ_H^1, and $\mathrm{Tr}(\mathbf{K})$ for all runs on the ResNet-32 model and the CIFAR-10 dataset .

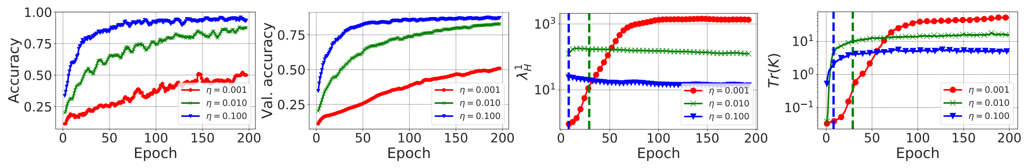


Figure 7: Additional figures for the experiments on ResNet-32 on CIFAR-10 dataset with different learning rates. From left to right: the evolution of accuracy, validation accuracy, λ_H^1 and $\mathrm{Tr}(\mathbf{K})$.

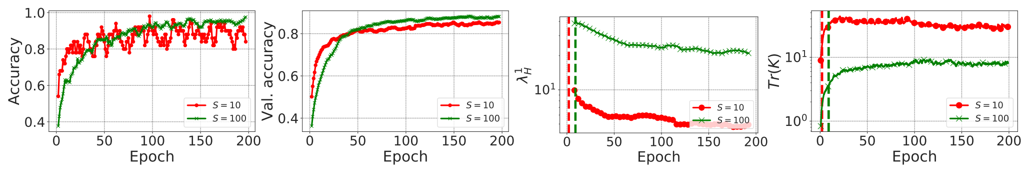


Figure 8: Additional figures for the experiments on ResNet-32 on CIFAR-10 dataset with different batch sizes. From left to right: the evolution of accuracy, validation accuracy, λ_H^1 and $\mathrm{Tr}(\mathbf{K})$.

[3]The model weights used can be found here: `https://tfhub.dev/google/bert_multi_cased_L-12_H-768_A-12/1`

SimpleCNN on CIFAR-10. In Fig. 7 and Fig. 8 we report accuracy on the training set and the validation set, λ_H^1, and $\mathrm{Tr}(\mathbf{K})$ for all runs on the SimpleCNN model and the CIFAR-10 dataset .

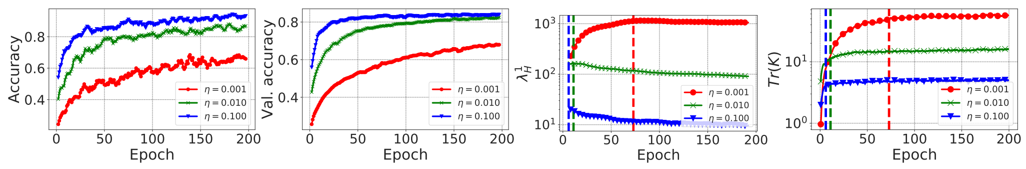


Figure 9: Additional figures for the experiments on SimpleCNN on CIFAR-10 dataset with different learning rates. From left to right: the evolution of accuracy, validation accuracy, λ_H^1 and $\mathrm{Tr}(\mathbf{K})$.

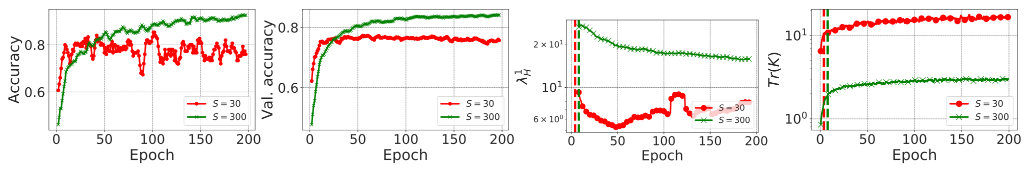


Figure 10: Additional figures for the experiments on SimpleCNN on CIFAR-10 dataset with different batch sizes. From left to right: the evolution of accuracy, validation accuracy, λ_H^1 and $\mathrm{Tr}(\mathbf{K})$.

LSTM on IMDB. In Fig. 11 and Fig. 12 we report accuracy on the training set and the validation set, λ_H^1, and $\mathrm{Tr}(\mathbf{K})$ for all runs on the LSTM model and the IMDB dataset.

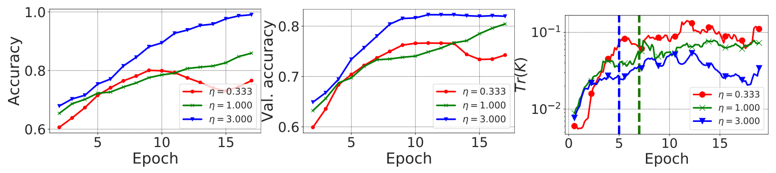


Figure 11: Additional figures for the experiments on LSTM on IMDB dataset with different learning rates. From left to right: the evolution of accuracy, validation accuracy, and $\mathrm{Tr}(\mathbf{K})$.

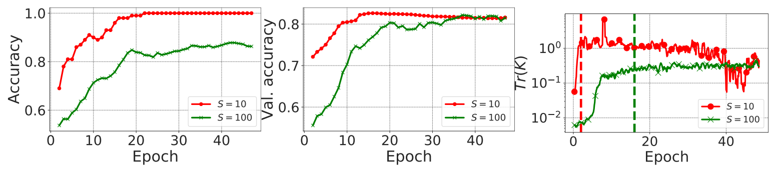


Figure 12: Additional figures for the experiments on LSTM on IMDB dataset with different batch sizes. From left to right: the evolution of accuracy, validation accuracy, λ_H^1 and $\mathrm{Tr}(\mathbf{K})$.

BERT on MNLI. In Fig. 13 and Fig. 14 we report results for the BERT model on the MNLI dataset.

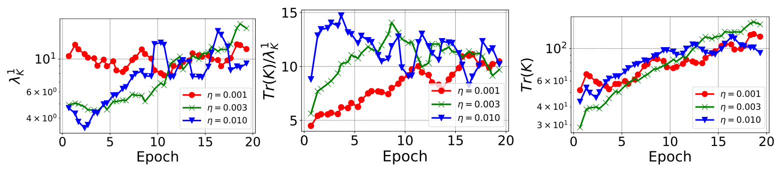


Figure 13: Results of experiment on the BERT model and MNLI dataset for different learning rates. From left to right: evolution of λ_K^1, $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$ and $\mathrm{Tr}(\mathbf{K})$.

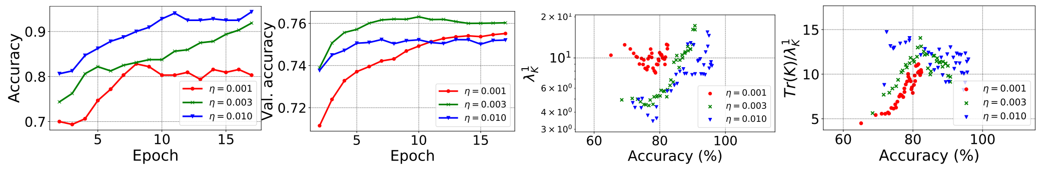


Figure 14: Evolution of accuracy and validation accuracy for experiment in Fig. 13. Each of the dots in the two rightmost plots represents a single iteration, on the horizontal axis we report the training accuracy, and on the vertical axis λ_K^1 or $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$.

MLP on FMNIST. In Fig. 15 and Fig. 16 we report results for the MLP model on the FMNIST dataset.

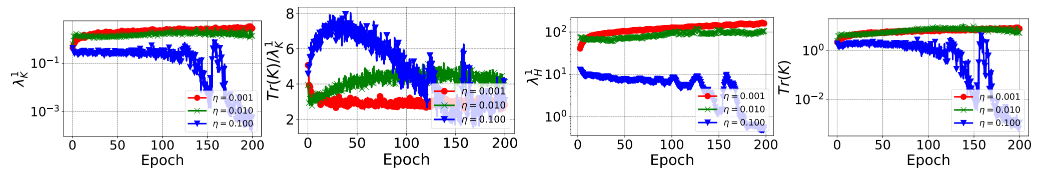


Figure 15: Results of experiment on the MLP model and FMNIST dataset for different learning rates. From left to right: evolution of λ_K^1, $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$, λ_H^1 and $\mathrm{Tr}(\mathbf{K})$.

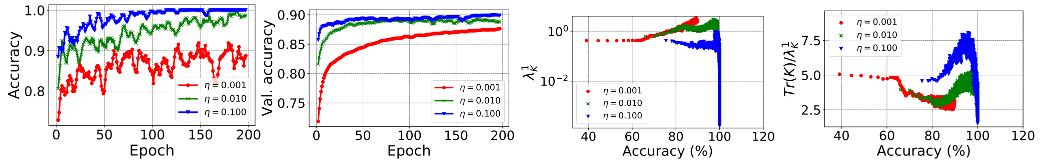


Figure 16: Same as Fig. 14 for the MLP model on FMNIST dataset.

DenseNet on ImageNet. In Fig. 17 and Fig. 18 we report results for the DenseNet-121 model on the ImageNet dataset.

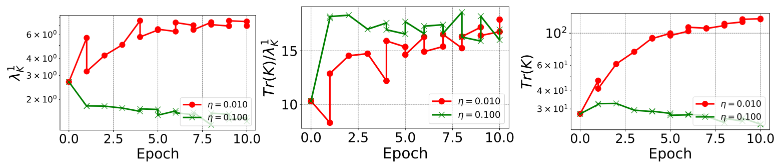


Figure 17: Results of experiment on the DenseNet model and ImageNet dataset for different learning rates. From left to right: evolution of λ_K^1, $\mathrm{Tr}(\mathbf{K})/\lambda_K^1$ and $\mathrm{Tr}(\mathbf{K})$.

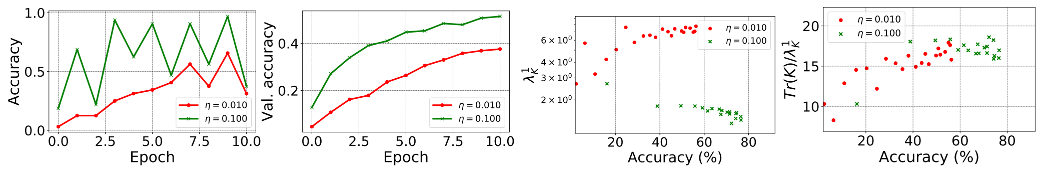


Figure 18: Same as Fig. 14 for the DenseNet model on ImageNet dataset.