A Walk with SGD

Chen Xing [* 1 2]   Devansh Arpit [* 2]   Christos Tsirigotis [3 2]   Yoshua Bengio [2 4]

Abstract

Exploring why stochastic gradient descent (SGD) based optimization methods train deep neural networks (DNNs) that generalize well has become an active area of research recently. Towards this end, we empirically study the dynamics of SGD when training over-parametrized deep networks. Specifically we study the DNN loss surface along the trajectory of SGD by interpolating the loss surface between parameters from consecutive *iterations* and tracking various metrics during the training process. We find that the covariance structure of the noise induced due to mini-batches is quite special that allows SGD to descend and explore the loss surface while avoiding barriers along its path. Specifically, our experiments show evidence that for the most part of training, SGD explores regions along a valley by bouncing off valley walls at a height above the valley floor. This 'bouncing off walls at a height' mechanism helps SGD traverse larger distance for small batch sizes and large learning rates which we find play qualitatively different roles in the dynamics. While a large learning rate maintains a large height from the valley floor, a small batch size injects noise facilitating exploration. We find this mechanism is crucial for generalization because the floor of the valley has barriers and this exploration above the valley floor allows SGD to quickly travel far away from the initialization point (without being affected by barriers) and find flatter regions, corresponding to better generalization.

1. Introduction

Deep neural networks (DNNs) trained with algorithms based on stochastic gradient descent (SGD) are able to tune the parameters of massively over-parametrized models to reach small training loss with good generalization despite the existence of numerous bad minima. This is especially surprising given DNNs are capable of overfitting random data with almost zero training loss (Zhang et al., 2016). This behavior has been studied by Arpit et al. (2017); Advani & Saxe (2017) where they suggest that deep networks generalize well because they tend to fit simple functions over training data before overfitting noise. It has been further discussed that model parameters that are in a region of flatter minima generalize better (Hochreiter & Schmidhuber, 1997; Keskar et al., 2016; Wu et al., 2017), and that SGD finds such minima when used with small batch size and high learning rate (Keskar et al., 2016; Jastrzębski et al., 2017; Smith et al., 2017; Chaudhari & Soatto, 2017). These recent papers frame SGD as a stochastic differential equation (SDE) under the assumption of using small learning rates. A main result of these papers is that the SDE dynamics remains the same as long as the ratio of learning rate to batch size remains unchanged. *However, this view is limited due to its assumption and ignores the importance of the structure of SGD noise (i.e., the gradient covariance) and the qualitative roles of learning rate and batch size, which remain relatively obscure.*

On the other hand, various variants of SGD have been proposed for optimizing deep networks with the goal of addressing some of the common problems found in the high dimensional non-convex loss landscapes (Eg. saddle points, faster loss descent etc). Some of the popular algorithms used for training deep networks apart from vanilla SGD are SGD with momentum (Polyak, 1964; Sutskever et al., 2013), AdaDelta (Zeiler, 2012), RMSProp (Tieleman & Hinton, 2012), Adam (Kingma & Ba, 2014) etc. However, for any of these methods, currently there is little theory proving they help in improving generalization in DNNs (which by itself is currently not very well understood), although there have been some notable efforts (Hardt et al., 2015; Kawaguchi et al., 2017). *This raises the question of whether optimization algorithms that are designed with the goal of solving the aforementioned high dimensional problems also help in finding minima that generalize well, or put differently, what attributes allow optimization algorithms to find such good minima in the non-convex setting.*

We take a step towards answering the above two questions in this paper for SGD (without momentum) through qualitative

---
[*]Equal contribution [1]College of Computer and Control Engineering, Nankai University, China [2]MILA, Université de Montréal [3]Aristotle University of Thessaloniki, Greece [4]CIFAR Senior Fellow. Correspondence to: Chen <xingchen1113@gmail.com>, Devansh <devansharpit@gmail.com>.

experiments. Specifically, we observe the following.

1. The structure of the SGD gradient covariance induced by mini-batches is quite essential towards both better optimization on the training set, as well as finding minima that generalize well. Conversely, artificially added isotropic noise in the absence of mini-batch induced stochasticity is bad for DNN optimization.

2. The loss landscape for classification tasks along SGD's path consists of *valleys* and the optimizer bounces off these valley walls at a height from the valley floor for most of the training, in search of flatter minima, and likely never crosses a *barrier*: we say a *barrier is crossed* when we see a point in the parameter space interpolated between the points just before and just after a single update step, such that the loss at the barrier point is higher than the loss at both of the other points.

3. Learning rate controls the height at which the optimization bounces above the valley floor while batch size controls the level of noise due to gradient covariance which seems to facilitate exploration of different regions in the valley. This is visible from the parameter distance from initialization. Conversely, in the absence of noise, the full batch gradient oscillates between valley walls without much exploration and the parameters do not move far away from initialization.

4. The floor of the valley seems to have barriers or rough terrain which may hinder exploration. Thus using a large learning rate helps avoid encountering these barriers by maintaining a large height (thus moving over the barriers instead of crossing them), while a small batch size facilitates exploration along the valley for finding wider regions. In this way, learning rate and batch size exhibit different qualitative roles in SGD dynamics.[1]

5. SGD roughly finds flatter regions as training progresses (measured using the Hessian of the loss)

In light of these findings, we discuss learning rate schedules for training DNNs, and that increasing the learning rate gradually during training might be helpful, although eventually annealing it is needed for convergence. We also discuss similarities and differences of our empirical findings in the context of classical optimization theory in the quadratic setting.

2. Background and Related Work

Various algorithms have been proposed in the deep learning community for optimizing deep neural networks, designed

---

[1]This implies that except when using a reasonably small learning rate (which would make the SDE approximation hold), the effect of small batch size with a certain learning rate cannot be achieved by using a large batch size with a proportionally large learning rate (observed by Goyal et al. (2017)).

from the view point of tackling various high dimensional optimization problems like oscillation during training (SGD with momentum (Polyak, 1964)), oscillations around minima (Nesterov momentum (Nesterov, 1983; Sutskever et al., 2013)), saddle points (Dauphin et al., 2014), automatic decay of the learning rate (AdaDelta (Zeiler, 2012), RMSProp (Tieleman & Hinton, 2012) and ADAM (Kingma & Ba, 2014)), etc.

On the other hand, as noted by Amari (1998) in the case of statistical models with non-euclidean Riemannian space, the direction of steepest descent[2] is not the gradient direction $\mathbf{g}$ but rather the direction $\mathbf{F}^{-1}\mathbf{g}$, where $\mathbf{F}$ is the Fisher information matrix which defines the metric of the Riemannian space. This line of work has been extended by others (Roux et al., 2008; Martens & Grosse, 2015) in the hope of making the original goal scalable for deep networks with a large number of parameters.

However, currently there is insufficient theory to understand what kind of minima generalize better and whether such minima can be reached by the aforementioned optimization algorithms under random initialization; although empirically it has been observed that wider minima (that can be quantified by low Hessian norm) seem to have better generalization (Keskar et al., 2016; Wu et al., 2017; Jastrzębski et al., 2017) due to their low complexity and are more likely to be reached under random initialization given their larger volumes (Wu et al., 2017). Moreover, there would be no hope of finding reasonably good minima if there was no structure in the loss landscape of DNNs and all minima were disconnected (separated by barriers), because in the worst case it would require the optimization algorithm to exhaustively search among all these minima. Fortunately this does not seem to be the case, and the general greedy approach of descending along the negative gradient with mini-batch induced stochasticity seems to do a reasonably good job at finding minima that generalize well. This is true even though these optimization algorithms have not been explicitly tailored towards exploiting the geometry of loss landscapes specific to DNNs.

These arguments raise the question of whether the intuitions behind the designs of the various optimization algorithms are really the reasons behind their success in deep learning or there are other underlying mechanisms that make them successful.

To understand this aspect better, a number of (mostly) recent papers study SGD as a stochastic differential process (Kushner & Yin, 2003; Mandt et al., 2017; Chaudhari & Soatto, 2017; Smith & Le, 2017; Jastrzębski et al., 2017; Li et al., 2015) under the assumption (among others) that the learning

---

[2]direction of largest change in objective for a unit change in parameters in terms of KL divergence

rate is reasonably small. Broadly, these papers show that the stochastic fluctuation in the stochastic differential equation simulated by SGD is governed by the ratio of learning rate to batch size. Hence according to this theoretical framework, the training dynamics of SGD should remain roughly identical when changing learning rate and batch size by the same factor. However, given DNNs (especially Resnet (He et al., 2016) like architectures) are often trained with quite large learning rates, the small learning rate assumption may be a pitfall of this theoretical framework[3]. But this theory is nonetheless useful since learning rates do attain small values during training due to annealing or adaptive scheduling, so this framework may indeed apply during parts of training. *In this paper we attempt to go beyond these analyses to study the different qualitative roles of the noise induced by large learning rate versus the noise induced by a small batch size.*

There have also been work that consider SGD as a diffusion process where SGD is running a Brownian motion in the parameter space. Li et al. (2017) hypothesize this behavior of SGD and theoretically show that this diffusion process would allow SGD to cross barriers and thus escape sharp local minima. The authors use this theoretical result to support the findings of Keskar et al. (2016) who find that SGD with small mini-batch find wider minima. Hoffer et al. (2017) on the other hand make a similar hypothesis based on the evidence that the distance moved by SGD from initialization resembles a diffusion process, and make a similar claim about SGD crossing barriers during training. *Contrary to these claims, we find that interpolating the loss surface traversed by SGD on a per iteration basis suggests SGD almost never crosses any significant barriers for most of the training.*

There is also a long list of work towards understanding the loss surface geometry of DNNs from a theoretical standpoint. Dotsenko (1995); Amit et al. (1985); Choromanska et al. (2015) show that under certain assumptions, the DNN loss landscape can be modeled by a spherical spin glass model which is well studied in terms of its critical points. Safran & Shamir (2016) show that under certain mild assumptions, the initialization is likely to be such that there exists a continuous monotonically decreasing path from the initial point to the global minimum. Freeman & Bruna (2016) theoretically show that for DNNs with rectified linear units (ReLU), the level sets of the loss surface become more connected as network over-parametrization increases. This has also been justified by Sagun et al. (2017) who show that the hessian of deep ReLU networks is degenerate when the network is over-parametrized and hence the loss surface is flat along such degenerate directions. Goodfellow et al. (2014) empirically show that the convex interpolation of

the loss surface from the initialization to the final parameters found by optimization algorithms do not cross any significant barriers. *Broadly these studies analyze DNN loss surfaces (either theoretically or empirically) in isolation from the optimization dynamics.*

In our work we do not study the loss surface in isolation, but rather analyze it through the lens of SGD. In other words, we study the DNN loss surface along the trajectory of SGD and track various metrics while doing so, from which we deduce both how the landscape relevant to SGD looks like, and how the hyperparameters of SGD, viz. learning rate and batch size, help SGD maneuver through it.

3. A Walk with SGD

We now begin our analysis of studying the loss surface of DNNs along the trajectory of optimization updates. Specifically, consider that the parameters θ of a DNN are initialized to a value θ_0. When using an optimization method to update these parameters, the tth update step takes the parameter from state θ_t to θ_{t+1} using estimated gradient $\mathbf{g}_t$ as,

$$\theta_{t+1} = \theta_t - \eta\mathbf{g}_t \qquad (1)$$

where η is the learning rate. Notice the tth update step implies the tth epoch *only* in the case when using the full batch gradient descent (GD– gradient computed using the whole dataset). In the case of stochastic gradient descent, one iteration is an update from gradient computed from a mini-batch. We then interpolate the DNN loss between the convex combination of θ_t and θ_{t+1} by considering parameter vectors $\theta_t^\alpha = (1-\alpha)\theta_t + \alpha\theta_{t+1}$, where $\alpha \in [0,1]$ is chosen such that we obtain 10 samples uniformly placed between these two parameter points. Simultaneously, we also keep track of two metrics– the cosine of the angle between two consecutive gradients $\cos(\mathbf{g}_{t-1}, \mathbf{g}_t) := \mathbf{g}_{t-1}^T\mathbf{g}_t/(\|\mathbf{g}_{t-1}\|_2\|\mathbf{g}_t\|_2)$, and the distance of the current parameter θ_t from the initialization θ_0 given by $\|\theta_t - \theta_0\|_2$. As it will become apparent in a bit, these two metrics along with the interpolation curve help us make deductions about how the optimization *interacts* with the loss surface during its trajectory.

We perform experiments on MNIST (Lecun & Cortes), CIFAR-10 (Krizhevsky, 2009) and a subset of the tiny Imagenet dataset (Russakovsky et al., 2015) using multi-layer perceptrons (MLP), VGG-11 (Simonyan & Zisserman, 2014) and Resnet-56 (He et al., 2016) architectures. Unless specified otherwise, for all experiments with VGG-11 we use a batch size of 100 and a fixed learning rate of 0.1 (to avoid any confounding factors from learning rate annealing). In the main text, we mostly show results for CIFAR-10 using the VGG-11 architecture due to space limitations. Experiments on all the other datasets and architectures can be found in the appendix. All the claims are consistent across datasets and architectures.

---

[3]For instance Goyal et al. (2017) investigate that increasing learning rate linearly with batch size helps to a certain extent but breaks down for very large learning rates.
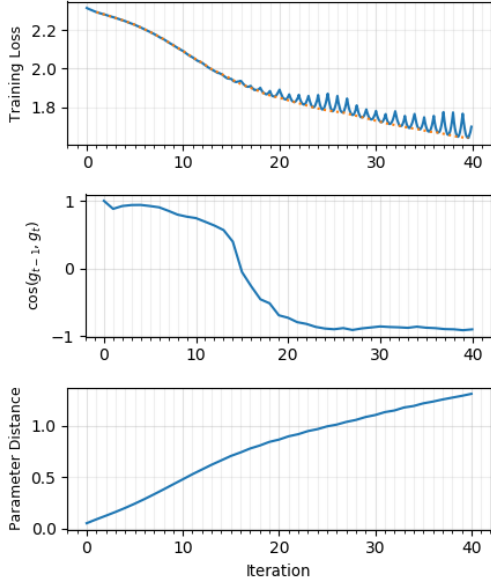
Figure 1. Plots for VGG-11 Epoch 1 trained using full batch **Gradient Descent** (GD) on CIFAR-10. **Top**: Training loss for the $1st$ 40 iterations of training. Between the training loss at every consecutive iteration, we uniformly sample 10 points between the parameters before and after a training update and calculate the loss at these points. Thus we take a slice of the loss surface between two iterations. These loss values are plotted between every consecutive training loss value from training updates. For instance, between iterations 20 and 21 (vertical gray lines), there are 10 loss values interpolating the loss surface. The dashed orange line connects the minimum of the loss interpolation between consecutive iterations (this minimum denotes the valley floor along the interpolation). **Middle**: Cosine of the angle between gradients from two consecutive iterations. **Bottom**: Parameter distance from initialization. **Gist**: The loss interpolation between consecutive iterations have a minimum for iterations where cosine is highly negative (close to -1 after around 20 iterations meaning the consecutive gradients are almost along opposite directions), suggesting the optimization is oscillating along the walls of a valley like structure. The valley floor (dashed orange line) reduces monotonously.

3.1. Optimization Trajectory

We first experiment with full batch gradient descent (GD) to study its behavior before jumping to the analysis of SGD to isolate the confounding factor of mini-batch induced stochasticity. The plot of training loss interpolation between consecutive iterations (referred in the figure as *training loss*), $\cos(\mathbf{g}_{t-1}, \mathbf{g}_t)$, and *parameter distance* $\|\theta_t - \theta_0\|_2$ for CIFAR-10 on VGG-11 architecture optimized using full batch gradient descent is shown in Figure 1 for the first 40 iterations of training. To be clear, the x-axis is calibrated by the number of iterations, and there are 10 interpolated loss values between each consecutive iterations (vertical gray lines) in the *training loss* plot which is as described above (the cosine and parameter distance plots do not have any interpolations). This figure shows that the interpolated loss
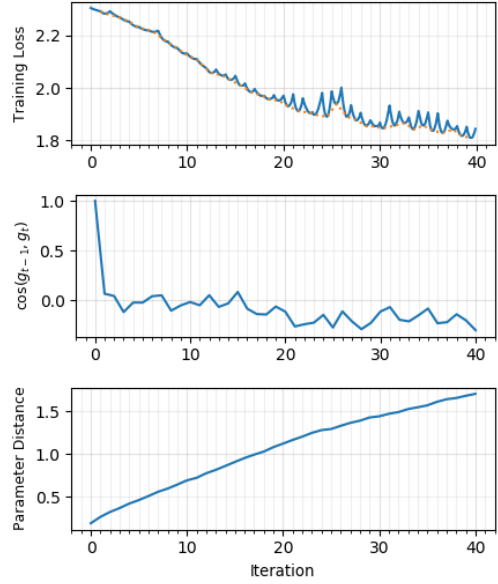
Figure 2. Plots for VGG-11 Epoch 1 trained using **SGD** on CIFAR-10. The descriptions of the plots are same as in Figure 1. **Gist**: The loss interpolation between consecutive iterations have a minimum for iterations and cosine is less negative compared with GD, suggesting the optimization is oscillating along the walls of a valley like structure but doing more exploration compared with GD. This is verified by the larger distance traveled by SGD compared with GD in Figure 1. The valley floor (dashed orange line) has many ups and downs showing barriers along SGD's path which do not affect its dynamics because SGD travels at a height above the floor.

between every consecutive parameters from GD optimization update after iteration 15 appears to be a quadratic-like structure with a minimum in between. Additionally, the cosine of the angle between consecutive gradients after iteration 15 is going negative and finally very close to -1, which means the consecutive gradients are almost along opposite directions. These two observations together suggest that the GD iterate is bouncing between walls of a valley-like landscape. For the iterations where there is a minimum in the interpolation between two iterations, we refer to this minimum as the *floor* of the valley (these valley floors are connected by dashed orange line in figure 1 for clarity). Thus we see that for GD, the floor is reducing almost monotonously. As we will see, this is not the case with SGD, and this GD behavior shows lack of exploration for better minima. Take note that the parameter distance from initialization during these 40 iterations reaches ~ 1.4.

Now we perform the same analysis for SGD. Notice that even though the updates are performed using mini-batches for SGD, the training loss values used in the plot are computed using the full dataset to visualize the actual loss landscape. We show these plots for epoch 1 (Figure 2) in the main text and epoch 25 (Figure 15) and epoch 100 (Figure

Table 1. Number of barriers crossed during training of a whole epoch (450 iterations) for VGG-11 and Resnet-56 on CIFAR-10 and MLP on MNIST. We say a barrier is crossed during a training update step if there exists a point interpolated between the parameters before and after an update which has a loss value higher than the loss at either points. For most parts of the training, we find that SGD does not cross any significant number of barriers.

	Epoch 1	Epoch 10	Epoch 25	Epoch 100
VGG-11	0	0	5	13
Resnet-56	0	0	2	23
MLP	0	3	5	-
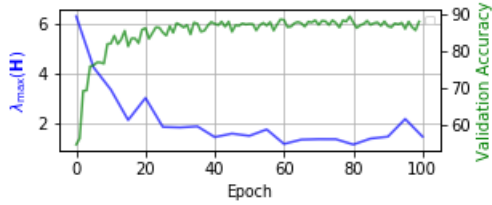
Figure 3. Max eigenvalue (spectral norm) of the Hessian and validation accuracy for VGG-11 trained on CIFAR-10 using a fixed learning rate of 0.1 and batch size 100. The spectral norm roughly decreases with training but starts increasing slightly towards the end. Similarly validation accuracy roughly improves throughout training but drops towards the end.

16) in the appendix. We find that there are many qualitative differences compared with the GD case. We see that the cosine of the angle between gradients from consecutive iterations are significantly less negative, suggesting that instead of oscillating in the same region, SGD is quickly moving away from its previous position. This can be verified by the parameter distance from initialization. We see that the distance after 40 iterations is ~ 1.7, which is larger than the distance moved by GD[4]. Finally and most interestingly, we see that the height of valley floor has many ups and downs for consecutive iterations in contrast with that of GD (emphasized by the dashed orange line in figure 2), which means that there is a rough terrain or barriers along the path of SGD that could hinder exploration if the optimization was traveling too close to the valley floor.

A similar analysis for Resnet-56 on CIFAR-10, MLP on MNIST, VGG-11 on tiny ImageNet trained using full batch GD for the first epoch are shown in Figures 9, 17,20 respectively in the appendix. The occurrence of valley-like structures between consecutive iterations along with negative cosine for two consecutive gradients are evident for all these architectures trained on different data sets. The same analysis for SGD for the first epoch are shown in Figures 10,18 and 21 in the appendix. For all architectures, SGD travels farther than full batch gradient descent and the valley floor for all cases have barriers.

Since we qualitatively only show the interpolations of a few iterations from each epoch for visual clarity, the claim about

---

[4]In general, after the same number of updates, GD traverses a smaller distance compared with SGD, see (Hoffer et al., 2017)

optimization not crossing barriers does not extend to the other iterations. So we quantitatively measure for the entire epoch if barriers are crossed. This result is shown in table 1 for VGG-11 and Resnet-56 trained on CIFAR-10 (trained for 100 epochs) and an MLP trained on MNIST (trained for 40 epochs). As we see, no barriers are crossed for most parts of the training. We further compute the number of barriers crossed for the first 40 epochs for VGG-11 on CIFAR-10 shown in Figure 8 in the appendix: no barriers are crossed for most of the epochs and even for the barriers that are crossed towards the end, we find that their heights are substantially smaller compared with the loss value at the corresponding point during training, meaning they are not significant.

Finally, we track the spectral norm of the Hessian along with the validation accuracy while the model is being trained[5]. This plot is shown in Figure 3 for VGG-11 (and Figure 7 for Resnet-56 in the appendix). We find that the spectral norm reduces as training progresses (hence SGD finds flatter regions) but starts increasing towards the end. This is mildly correlated with a drop in validation accuracy towards the end. Although we expect the trends of these two plots to be roughly inversely proportional (which is true), we do not hope for a very strong correlation between them because apart from the recent empirical findings suggesting flat minima generalizing better (Hoffer et al., 2017; Jastrzębski et al., 2017), there is currently little theory behind this claim and in fact theoretical arguments which seem in contradiction with it (Dinh et al., 2017). Additionally, the spectral norm only captures the largest eigenvalue of the Hessian. Note that we do not track the Frobenius norm as it can be misleading because the Hessian may have negative eigenvalues and the Frobenius norm sums the square of all eigenvalues.

3.2. Qualitative Roles of Learning Rate and Batch Size

We now focus in more detail on how the learning rate and batch size play qualitatively different roles during SGD optimization. As an extreme case, we already saw in the last section that when using GD vs SGD, the cosine of the angle between gradients from two consecutive iterations $\cos(\mathbf{g}_{t-1}, \mathbf{g}_t)$ is significantly closer to -1 (180 degrees) in the case of GD in contrast with SGD. Now we show on a more granular scale that changing the batch size gradually (keeping the learning rate fixed) changes $\cos(\mathbf{g}_{t-1}, \mathbf{g}_t)$, while changing the learning rate gradually (keeping batch size fixed) does not. It is shown in Figure 4 for VGG-11 trained on CIFAR-10. Notice the cosine is significantly more negative for larger batch sizes, implying that for larger batch sizes, the optimization is bouncing more within the

---

[5]Note that we track the spectral norm in the train mode of batchnorm; we observe that in validation mode the values are significantly larger. Tracking the value in train mode is fair because this is what SGD *experiences* during training.
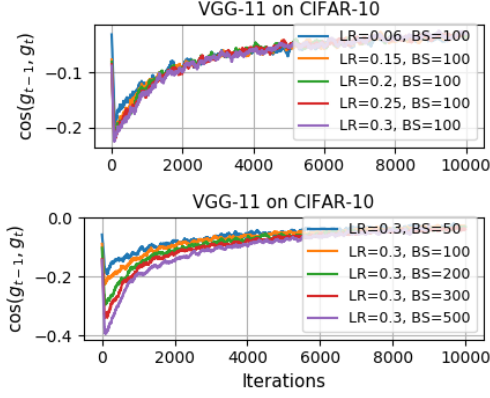
Figure 4. Changing batch size changes the cosine of angle between consecutive gradients while changing learning rate does not have any significant effect on the cosine. This shows batch size has a qualitatively different role compared with learning rate. Note that the curves are smoothened for visual clarity.

Table 2. Average height of SGD above valley floor across the iterations in one epoch for different epochs during the training of VGG-11 and Resnet-56 using SGD on CIFAR-10. Here the height at iteration t is defined defined by $\frac{\mathcal{L}(\theta_t)+\mathcal{L}(\theta_{t+1})-2\mathcal{L}(\theta_t^{\min})}{2}$.

VGG-11	Epoch 1	Epoch 10	Epoch 25	Epoch 100
LR 0.1	0.0625	0.0199	0.0104	0.0025
LR 0.05	0.0102	0.0050	0.0035	0.0011
Resnet-56	Epoch 1	Epoch 10	Epoch 25	Epoch 100
LR 0.3	0.0380	0.0131	0.0094	0.0017
LR 0.15	0.0084	0.0034	0.0020	0.0013

same region instead of traveling farther along the valley as the case of small batch sizes. This behavior is verified by the smaller distance of parameters from initialization during training for larger batch size which is also discussed by Hoffer et al. (2017). This suggests that the noise from a small mini-batch size facilitates exploration that may lead to better minima and that this is hard to achieve by changing the learning rate.

On the other hand, we find that the learning rate controls the height from the valley floor at which the optimization oscillates along the valley walls which is important for avoiding barriers along SGD's path. Specifically, to quantify the height at which the optimization is bouncing above the valley floor, we make the following computations. Suppose at iterations t and $t+1$ of training, the parameters are given by θ_t and θ_{t+1} respectively, and from the 10 points sampled uniformly between θ_t and θ_{t+1} given by $\theta_t^\alpha = (1-\alpha)\theta_t + \alpha\theta_{t+1}$ for different values of $\alpha \in [0, 1]$, we define $\theta_t^{\min} := \theta_t^{\arg\min_\alpha \mathcal{L}(\theta_t^\alpha)}$, where $\mathcal{L}(\theta)$ denotes the DNN loss at parameter θ using the whole training set. Then we define the height of the iterate from the valley floor at iteration t as $\frac{\mathcal{L}(\theta_t)+\mathcal{L}(\theta_{t+1})-2\mathcal{L}(\theta_t^{\min})}{2}$. We then separately compute the average height for all iterations of epochs 1, 10, 25 and 100. These values are shown in table 2 for VGG-11
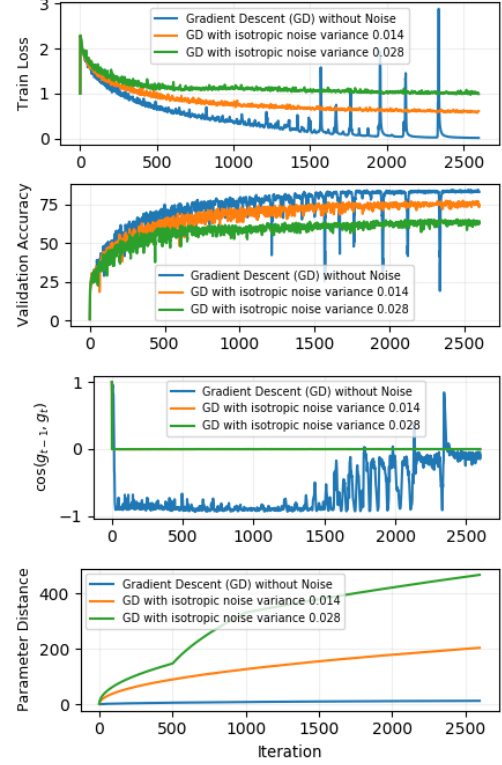

Figure 5. Plots for VGG-11 trained by GD (without noise) and GD with artificial isotropic noise sampled from Gaussian distribution with different variances. Models trained using GD with added isotropic noise get stuck in terms of training loss and have worse validation performance compared with the model trained with GD.

and Resnet-56 architectures trained on CIFAR-10. They show that for almost all epochs, a smaller learning rate leads to a smaller height from the valley floor. Since the valley floor has barriers, it would increase the risk of hindering exploration for flatter minima. This has been corroborated by the recent empirical observations that smaller learning rates lead to sharper minima and poor generalization (Smith et al., 2017; Jastrzębski et al., 2017).

The same experiment verifying the roles of learning rate and batch size is also conducted on Resnet-56 with CIFAR-10, MLP with MNIST and VGG-11 with tiny ImageNet. Plots are shown in Figures 22, 23 and 24 respectively in the appendix and the conclusions we achieved and discussed above are consistent for all the tested architectures.

4. Importance of SGD Noise Structure

The gradient $\mathbf{g}_{SGD}(\theta)$ from mini-batch SGD at a parameter value θ is expressed as,

$$\mathbf{g}_{SGD}(\theta) = \mathbf{g}(\theta) + \frac{1}{\sqrt{B}}\mathbf{n}(\theta) \qquad (2)$$

where $\mathbf{n}(\theta) \sim \mathcal{N}(0, \mathbf{C}(\theta))$, $\mathbf{g}(\theta)$ denotes the expected gradient using all training samples, B is the mini-batch size and

$\mathbf{C}(\theta)$ is the gradient covariance matrix at θ. In the previous section we discussed how mini-batch induced stochasticity plays a crucial role in SGD based optimization. This stochasticity due to SGD has historically been attributed to helping the optimization escape local minima in DNNs during training. However, the importance of the structure of the gradient covariance matrix $\mathbf{C}(\theta)$ is often neglected in these claims. To better understand its importance, we study the training dynamics of full batch gradient with artificially added isotropic noise. Specifically, we treat isotropic noise as our null hypothesis to confirm that the structure of noise induced by mini-batches in SGD is important.

In our experiments, we first train our models with gradient descent (GD). Thus there is no noise sampled from the stochastic process of SGD during training. For gradient descent with isotropic noise experiment, we add isotropic noise at every iteration on top of the full gradient. The noise is sampled from a normal distribution with variance calculated by multiplying the maximum gradient variance of the model at the initialization with a factor of 0.1 and 0.05. We train all models until their training losses saturate and we monitor training loss, validation accuracy, the cosine of the angle between gradients from two consecutive iterations and the parameter distance from initialization.

We conduct this experiment using VGG-11 and Resnet-56 architectures for CIFAR-10, and MLP for MNIST using batch size 100 and a fixed learning rate of 0.1. Figure 5 shows the results for VGG-11. From the training loss and validation accuracy curves, we can see that adding even a small isotropic noise makes both the convergence[6] and generalization worse compared with the model trained with GD. The cosine of the angle between gradients of two consecutive iterations is close to -1 for 1500 iterations for GD, which means two consecutive gradients are almost along opposite directions. It is an extra evidence that GD makes the optimization bounce off valley walls, which is what we discussed in section 3. The parameter distance from initialization shows that models trained with isotropic noise travel farther away compared with the model trained using noiseless GD. These distances are much larger even compared with models trained with mini-batch SGD (not shown here) for the same number of updates. To gain more intuitions into this behavior, we also calculate the norm of the final parameters found by GD, SGD and the isotropic noise cases. The parameter norms are 87 and 82 for GD and SGD respectively, and 369 and 443 for the 0.014 and 0.028 isotropic variance case. These numbers corroborate the generally discussed notion that SGD finds solutions with small ℓ^2 norm (Zhang et al., 2016) compared with

---

[6]We additionally find that the model trained with isotropic noise gets stuck because we find that neither reducing learning rate, nor switching to GD at this point leads to reduction in training loss. However, switching to SGD makes the loss go down.

GD, and the fact that isotropic noise solutions have much larger norms and get stuck suggests that isotropic noise both hinders optimization and is bad for generalization. Results for Resnet-56 on CIFAR-10 and MLP on MNIST are shown in Figure 25 and Figure 26 in the appendix and they show the same phenomenon.
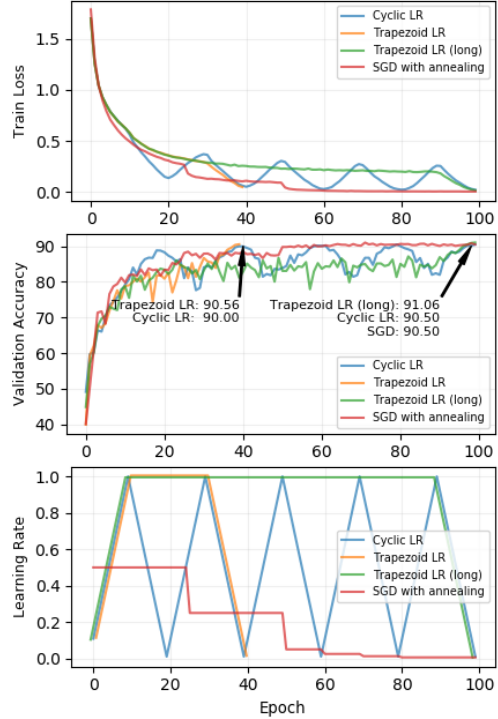

Figure 6. Plots for VGG-11 on CIFAR-10 trained using Cyclic learning rate (CLR), SGD with stepwise annealing, and trapezoid schedule. Cycles in the CLR schedule are redundant, which is shown by the trapezoid schedule.

Neelakantan et al. (2015) suggest adding isotropic noise to gradients and report performance improvement on a number of tasks. However, notice the crucial difference between our claim in this section and their setup is that the isotropic noise in their case is added on top of the noise due to the mini-batch induced stochasticity, while our experiments show that adding isotropic noise when using full dataset gradient (hence no noise is sampled from the gradient covariance matrix) makes both optimization harder (by getting stuck) and generalization worse, even when compared with full dataset gradient descent (without any noise), which is well known to be worse compared with generalization from SGD with small mini-batches (Keskar et al., 2016). Besides, Neelakantan et al. (2015) do not compare the performance of the additive isotropic noise with SGD with different batch sizes as a baseline, which we believe is important.

To gains insights into why the noise sampled from the gradient covariance matrix $\mathbf{C}(\theta)$ helps SGD, we note that there is a relationship between the covariance $\mathbf{C}(\theta)$ and the Hessian

$\mathbf{H}(\theta)$ of the loss surface at parameter θ which is revealed by the generalized Gauss Newton decomposition (see Sagun et al. (2017)) when using the cross-entropy (or negative log likelihood) loss. Let $p_i(\theta)$ denote the predicted probability output (of the correct class in the classification setting for instance) of a DNN parameterized by θ for the ith data sample (in total N samples). Then the negative log likelihood loss for the ith sample is given by, $\mathcal{L}_i(\theta) = -\log(p_i(\theta))$. *The relation between the Hessian* $\mathbf{H}(\theta)$ *and the gradient covariance matrix* $\mathbf{C}(\theta)$ *for the negative log likelihood case is given as*,

$$\mathbf{H}(\theta) = \mathbf{C}(\theta) + \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}_i(\theta)}{\partial p_i(\theta)} \cdot \frac{\partial^2 p_i(\theta)}{\partial \theta^2} \quad (3)$$

The derivation can be found in section D of the appendix. Thus we find that the Hessian and covariance at any point θ are closely related, and are almost equal near minima where the second term tends to zero. Although we do not empirically confirm this, the above relationship may imply that the mini-batch induced noise is roughly aligned with sharper directions of the loss landscape. This would prevent the optimization from converging along such directions unless a wider region is found, which could serve as an explanation for why SGD finds wider minima without relying on the framework of stochastic differential process (previous work) which assumes a reasonably small learning rate. We leave this for future work.

5. Learning Rate Schedule

We observe from table 2 that the optimization oscillates at a lower height as training progresses (which is likely because SGD finds flatter regions as training progresses, see Figure 3). As we discussed based on Figure 2, the floor of the DNN valley is highly non-linear with many barriers. Based on these two observations, it seems that it should be advantageous for SGD to maintain a large height from the floor of the valley to facilitate further exploration without getting hindered by barriers as it may allow the optimization to find flatter regions. Hence, this line of thought suggests that we should increase the learning rate as training progresses (of course eventually it needs to be annealed for convergence to a minimum). Recently (Smith & Topin, 2017) propose a cyclical learning rate (CLR) schedule which partially has this property. It involves linearly increasing the learning rate every iteration until a certain number of iterations, then similarly linearly reducing it, and repeat this process in a cycle. We now empirically show that multiple cycles of CLR are redundant, and simply increasing the learning rate until a certain point, and then annealing it leads to similar or better performance. Specifically, to rule out the need for cycles, as a null hypothesis, we increase the learning rate as in the first cycle of CLR, then keep it flat, then linearly anneal it (we call it the *trapezoid schedule*). For fairness, we

also plot the widely used step-wise learning rate annealing schedule. *In our experiments, we find that methods which increase learning rate during training may be considered slightly better*. The learning curves are shown in figures 6 in main text and 27 in appendix (with other details). We leave an extensive study of learning rate schedule design based on the proposed guideline as future work.

6. Discussion

We presented qualitative results to understand how GD and SGD interact with the DNN loss surface and avoided assumptions in order to rely instead on empirical evidence. We now draw similarities and differences between the DNN optimization dynamics that we have empirically found, with optimization theory in known settings that exhibit similar behavior (see section 5 of LeCun et al. (1998)). Based on our empirical evidence, we deduce that both GD and SGD move in a valley like landscape by bouncing off valley walls. This is reminiscent of optimization in a quadratic loss setting with a non-isotropic positive semi-definite Hessian, where the optimal learning rate η causes under-damping without divergence along eigenvectors of the Hessian which have eigenvalues λ_i such that $\lambda_i^{-1} < \eta < 2\lambda_i^{-1}$. On the other hand, in the case of DNNs trained with GD, we find that even though the training loss oscillates between valley walls during consecutive iterations, the floor of the valley decreases smoothly (see Figure 1 for instance). This is similar to the quadratic loss optimization with over-damped convergence along the eigenvectors corresponding to eigenvalues λ_i such that $\eta < \lambda_i^{-1}$.

On a different note, it is commonly conjectured that when training DNNs, SGD crosses barriers to escape local minima. Contrary to this commonly held intuition, we find that SGD almost never crosses any significant barriers along its path. More interestingly, when training with a certain learning rate (see figure 2), we find barriers at the floor of the valley but SGD avoids them by traveling at a height above the floor (due to large learning rate). Hence if we use a small learning rate, SGD should encounter such barriers and likely cross them. But we found in our experiments that this was not the case. This suggests that while in theory SGD is capable of crossing barriers (due to stochasticity), it does not do so because probably there exist other directions in such regions along which SGD can continue to optimize without crossing barriers. But since small learning rates empirically correlate with bad generalization, this suggests that moving over such barriers instead of crossing them by using a large learning rate is a good mechanism for exploration for good regions.

Finally, much of what we have discussed is based on the loss landscape of specific datasets and architectures along with network parameterization choices like rectified linear activation units (ReLUs) and batch normalization (Ioffe & Szegedy, 2015). These conclusions may differ for differ-

ent choices of architectures, network parametrization and datasets. In these cases analysis similar to ours can be performed to see if similar dynamics hold or not. Studying these dynamics may provide more practical guidelines for setting optimization hyperparameters.

Acknowledgments

References

Advani, Madhu S and Saxe, Andrew M. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

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

Amit, Daniel J, Gutfreund, Hanoch, and Sompolinsky, Haim. Spin-glass models of neural networks. *Physical Review A*, 32(2):1007, 1985.

Arpit, Devansh, Jastrzębski, Stanisław, Ballas, Nicolas, Krueger, David, Bengio, Emmanuel, Kanwal, Maxinder S, Maharaj, Tegan, Fischer, Asja, Courville, Aaron, Bengio, Yoshua, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242, 2017.

Chaudhari, Pratik and Soatto, Stefano. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. *arXiv preprint arXiv:1710.11029*, 2017.

Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, Gérard Ben, and LeCun, Yann. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pp. 192–204, 2015.

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

Dinh, Laurent, Pascanu, Razvan, Bengio, Samy, and Bengio, Yoshua. Sharp minima can generalize for deep nets. *arXiv e-prints*, 1703.04933, March 2017.

Dotsenko, Viktor. *An introduction to the theory of spin glasses and neural networks*, volume 54. World Scientific, 1995.

Freeman, C Daniel and Bruna, Joan. Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540*, 2016.

Goodfellow, Ian J, Vinyals, Oriol, and Saxe, Andrew M. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

Goyal, Priya, Dollár, Piotr, Girshick, Ross, Noordhuis, Pieter, Wesolowski, Lukasz, Kyrola, Aapo, Tulloch, Andrew, Jia, Yangqing, and He, Kaiming. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Hardt, Moritz, Recht, Benjamin, and Singer, Yoram. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

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

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

Hoffer, Elad, Hubara, Itay, and Soudry, Daniel. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1729–1739, 2017.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456, 2015.

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

Kawaguchi, Kenji, Kaelbling, Leslie Pack, and Bengio, Yoshua. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

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

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, Alex. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

Kushner, Harold and Yin, G George. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

Lecun, Yann and Cortes, Corinna. The MNIST database of handwritten digits. URL http://yann.lecun.com/exdb/mnist/.

LeCun, Yann, Bottou, Léon, Orr, Genevieve B, and Müller, Klaus-Robert. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 1998.

Li, Chris Junchi, Li, Lei, Qian, Junyang, and Liu, Jian-Guo. Batch size matters: A diffusion approximation framework on nonconvex stochastic gradient descent. *arXiv preprint arXiv:1705.07562*, 2017.

Li, Qianxiao, Tai, Cheng, et al. Stochastic modified equations and adaptive stochastic gradient algorithms. *arXiv preprint arXiv:1511.06251*, 2015.

Mandt, Stephan, Hoffman, Matthew D, and Blei, David M. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

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

Neelakantan, Arvind, Vilnis, Luke, Le, Quoc V, Sutskever, Ilya, Kaiser, Lukasz, Kurach, Karol, and Martens, James. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

Nesterov, Yurii. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pp. 372–376, 1983.

Polyak, Boris T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Roux, Nicolas L, Manzagol, Pierre-Antoine, and Bengio, Yoshua. Topmoumoute online natural gradient algorithm. In *Advances in neural information processing systems*, pp. 849–856, 2008.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Safran, Itay and Shamir, Ohad. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, pp. 774–782, 2016.

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

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

Smith, Leslie N and Topin, Nicholay. Super-convergence: Very fast training of residual networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2017.

Smith, Samuel L and Le, Quoc V. Understanding generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Smith, Samuel L, Kindermans, Pieter-Jan, and Le, Quoc V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Sutskever, Ilya, Martens, James, Dahl, George, and Hinton, Geoffrey. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Wu, Lei, Zhu, Zhanxing, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

Zeiler, Matthew D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.
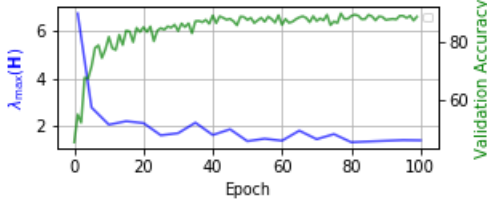
Figure 7. Max eigenvalue (spectral norm) of the Hessian and validation accuracy for Resnet-56 trained on CIFAR-10 using a fixed learning rate of 0.3 and batch size 100. The spectral norm roughly decreases with training but starts increasing slightly towards the end. Similarly validation accuracy roughly improves throughout training but drops towards the end.

Appendix

A. Optimization Trajectory

This is a continuation of section 3.1 in the main text. Here we show further experiments on other datasets and architectures.For all experiments with Resnet-56 we use a batch size of 100 and a fixed learning rate of 0.1 The analysis of GD training for Resnet-56 on CIFAR-10, MLP on MNIST and VGG-11 on tiny ImageNet are shown in figures 9, 17 and 20 respectively. Similarly, the analysis of SGD training for Resnet-56 on CIFAR-10 dataset for epochs 1, 2, 25 and 100 are shown in figures 10, 11, 12 and 13 respectively. The analysis of SGD training for VGG-11 on CIFAR-10 on epochs 2, 25,100 are shown in figures 14, 15 and 16. The analysis of SGD training for MLP on MNIST for epochs 1 and 2 are shown in figures 18 and 19. The analysis of SGD training for VGG-11 on tiny ImageNet for epochs 1 is shown in figure 21. The observations and rules we discovered and described in section 3 are all consistent for all these experiments. Specifically, for the interpolation of SGD for VGG-11 on tiny ImageNet, the valley-like trajectory is weird-looking but even so, according to our quantitative evaluation there is no barrier between any two consecutive iterations.

We track the spectral norm of the Hessian along with the validation accuracy while the model is being trained. This is shown in figure 7 for Resnet-56 trained on CIFAR-10.

B. Qualitative Roles of Learning Rate and Batch Size

This is a continuation of section 3.2 in the main text. In this section we show further experiments for the analysis of different roles of learning rate and batch size during training on various architectures and data sets. Figures 22, 23 and 24 shows the results for Resnet-56 on CIFAR-10, MLP on MNIST and VGG-11 on Tiny-Imagenet. In all of the experiments, training the model with smaller batch size will make the angle between gradients of two consecutive iterations larger, which means for smaller batch size, instead of

oscillating within the same region, the optimization travels farther along the valley, as we described in section 3.2. For all architectures, changing learning rate doesn't change the angles.

C. Learning Rate Schedule

This is a continuation of section 5 in which we show the other experiment we have done for learning rate schedule. We run the same experiment as described in section 5 on Resnet-56 with CIFAR-10 and it shows the rule for CLR, trapezoid schedule and SGD with stepwise annealing. Plots can be seen at figure 27. All schedules are tuned to their best performance with a hyperparameter grid search. For both Resnet-56 and VGG-11, we use batch size 100 for all models. The learning rate schedules are apparent from the figures themselves.

D. Importance of SGD Noise Structure

Here we derive in detail the relation between the Hessian and gradient covariance using the fact that for the negative log likelihood loss, $\frac{\partial \mathcal{L}_i(\theta)}{\partial p_i(\theta)} = -\frac{1}{p_i(\theta)}$, and $\frac{\partial^2 \mathcal{L}_i(\theta)}{\partial p_i(\theta)^2} = \frac{1}{p_i^2(\theta)}$.

$$\mathbf{H}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial^2 \mathcal{L}_i(\theta)}{\partial \theta^2} \tag{4}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \frac{\partial^2 \mathcal{L}_i(\theta)}{\partial p_i(\theta)^2} \cdot \frac{\partial p_i(\theta)}{\partial \theta} \frac{\partial p_i(\theta)}{\partial \theta}^T$$
$$+ \frac{\partial \mathcal{L}_i(\theta)}{\partial p_i(\theta)} \cdot \frac{\partial^2 p_i(\theta)}{\partial \theta^2} \tag{5}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\partial \mathcal{L}_i(\theta)}{\partial p_i(\theta)} \right)^2 \cdot \frac{\partial p_i(\theta)}{\partial \theta} \frac{\partial p_i(\theta)}{\partial \theta}^T$$
$$+ \frac{\partial \mathcal{L}_i(\theta)}{\partial p_i(\theta)} \cdot \frac{\partial^2 p_i(\theta)}{\partial \theta^2} \tag{6}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \mathcal{L}_i(\theta)}{\partial \theta} \frac{\partial \mathcal{L}_i(\theta)}{\partial \theta}^T + \frac{\partial \mathcal{L}_i(\theta)}{\partial p_i(\theta)} \cdot \frac{\partial^2 p_i(\theta)}{\partial \theta^2} \tag{7}$$

$$= \mathbf{C}(\theta) + \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \mathcal{L}_i(\theta)}{\partial p_i(\theta)} \cdot \frac{\partial^2 p_i(\theta)}{\partial \theta^2} \tag{8}$$

E. Discussion

In the main text, we talk about converge in the quadratic setting depending on the value of learning rate relative to the largest eigenvalue of the Hessian. The convergence in this setting has been visualized in 28.
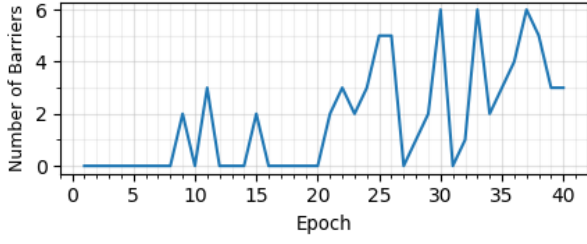
Figure 8. Numbers of barriers found during training loss interpolation for every epoch(450 iterations) for VGG-11 on CIFAR-10. We say a barrier exists during a training update step if there exists a point between the parameters before and after an update which has a loss value higher than the loss at either points. Note that even for these barriers, their heights (defined by $\frac{\mathcal{L}(\theta_t) + \theta_{t+1}) - 2\mathcal{L}(\theta_t^{\min})}{2}$) are substantially smaller compared with the value of loss at the corresponding iterations (not mentioned here), meaning they are not significant barriers.
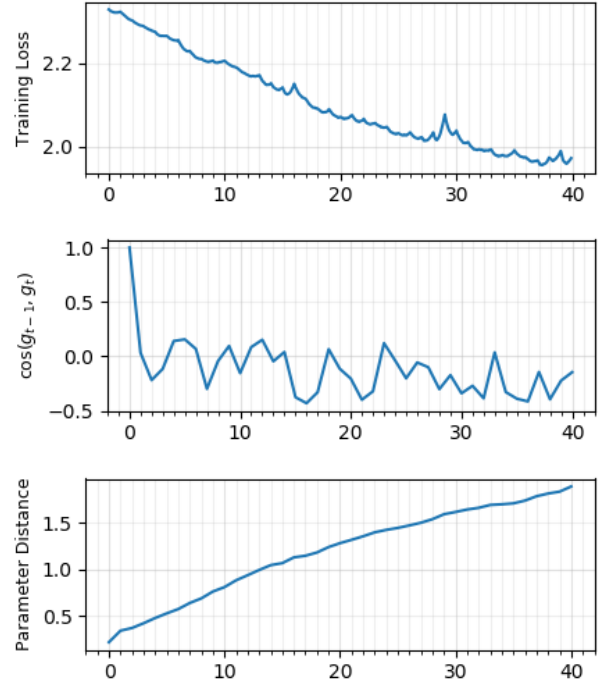
Figure 10. Plots for Resnet-56 Epoch 1 trained using **SGD** on CIFAR-10. The descriptions of the plots are same as in figure 1.
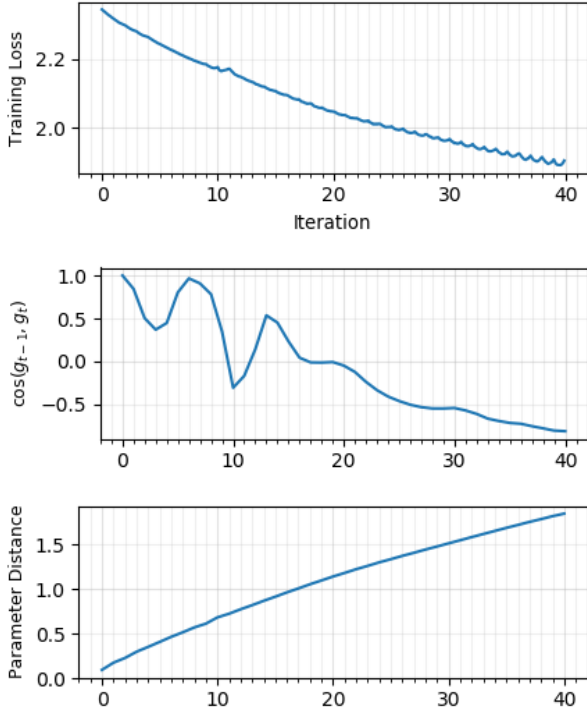
Figure 9. Plots for Resnet-56 Epoch 1 trained using full batch **Gradient Descent (GD)** on CIFAR-10. The descriptions of the plots are same as in figure 1.
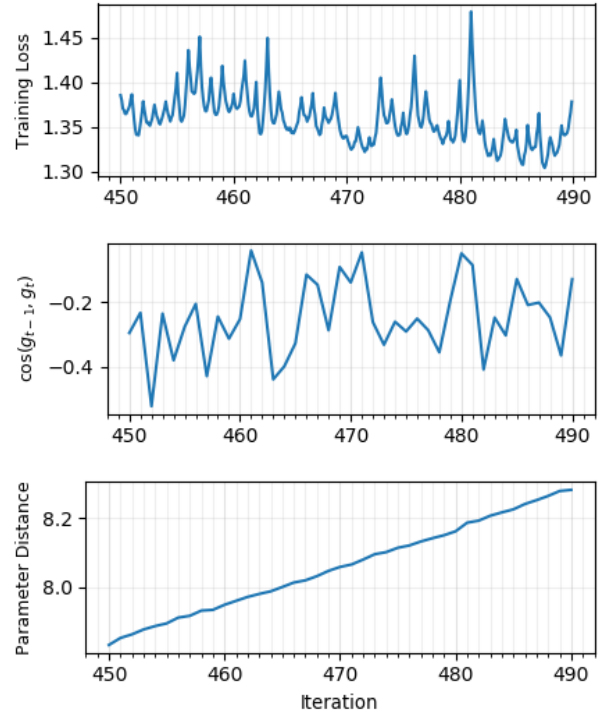
Figure 11. Plots for Resnet-56 Epoch 2 trained using **SGD** on CIFAR-10. The descriptions of the plots are same as in figure 1.
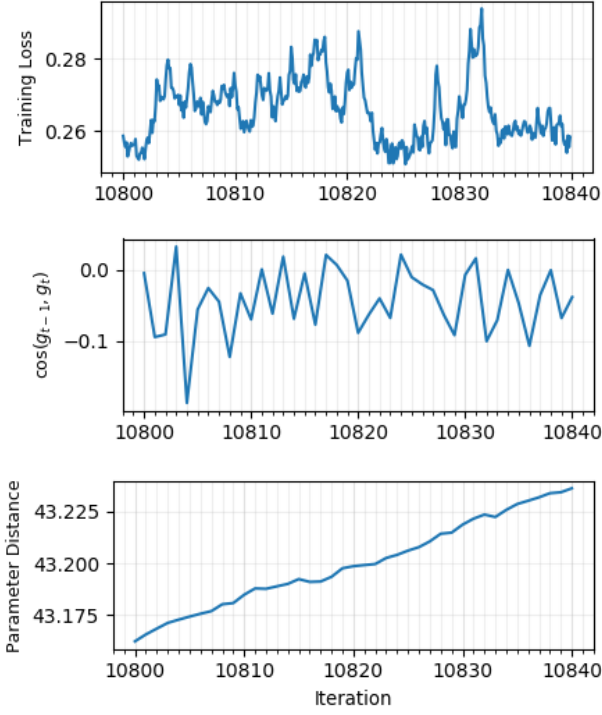
Figure 12. Plots for Resnet-56 Epoch 25 trained using **SGD** on CIFAR-10. The descriptions of the plots are same as in figure 1.
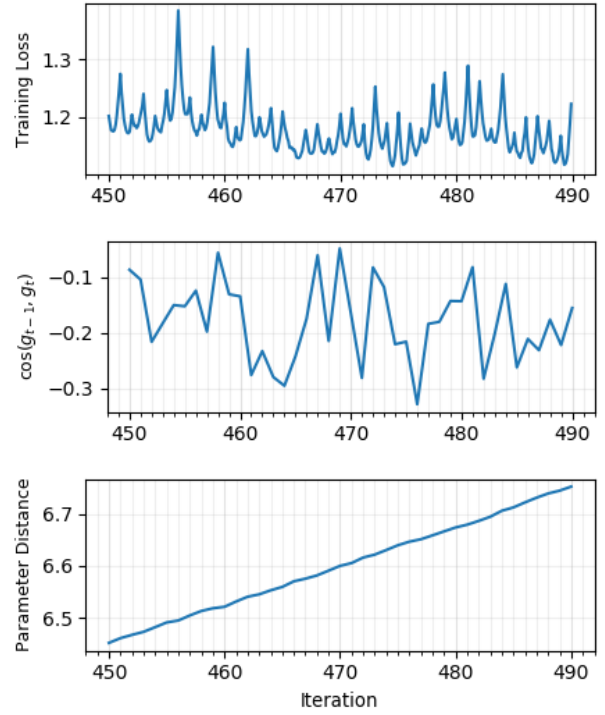
Figure 14. Plots for VGG-11 Epoch 2 trained using **SGD** on CIFAR-10. The descriptions of the plots are same as in figure 1.
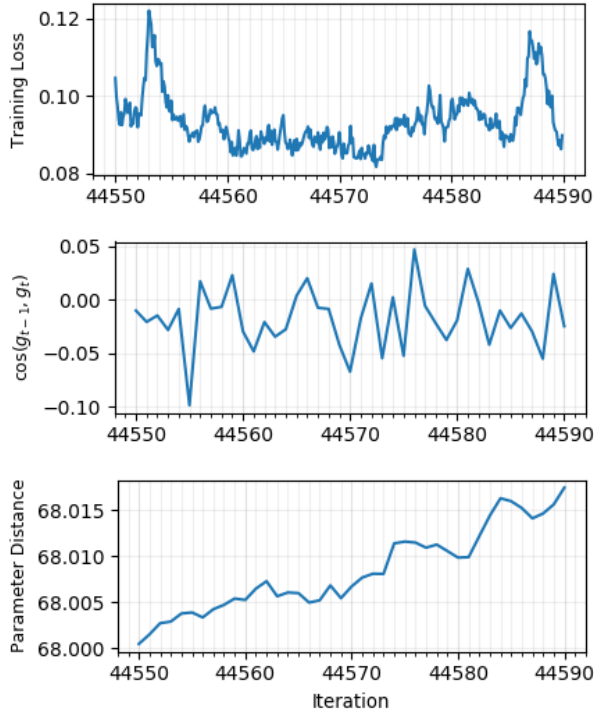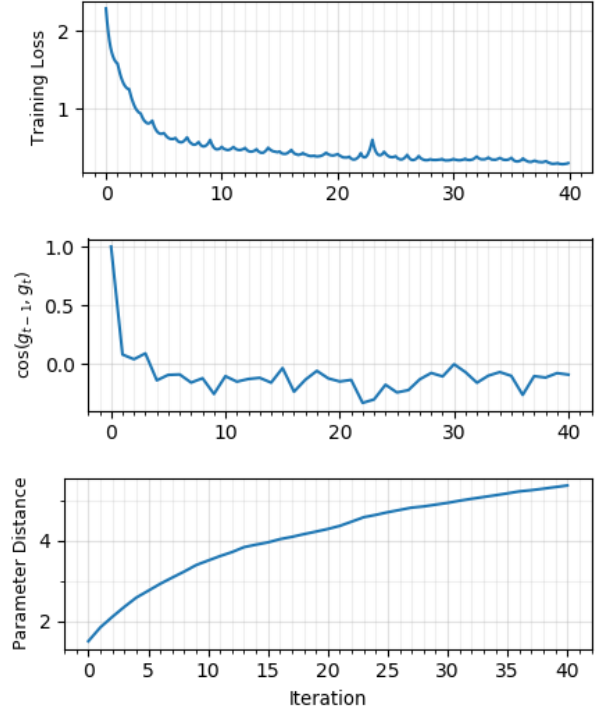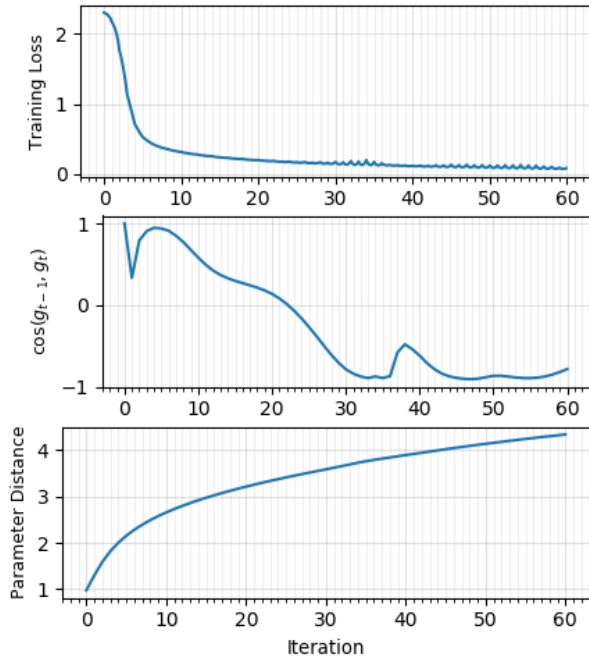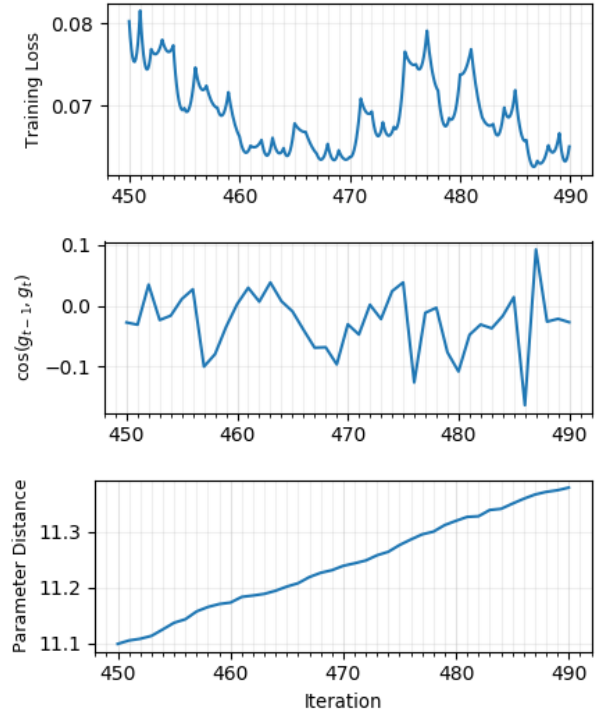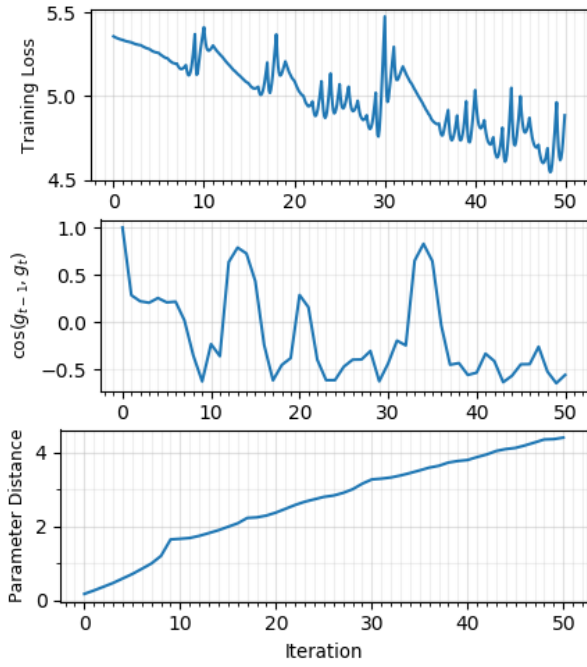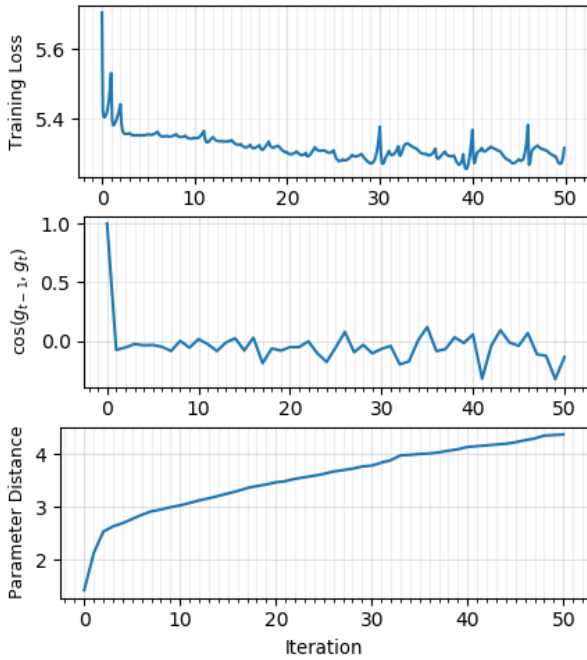

Figure 13. Plots for Resnet-56 Epoch 100 trained using **SGD** on CIFAR-10. The descriptions of the plots are same as in figure 1.
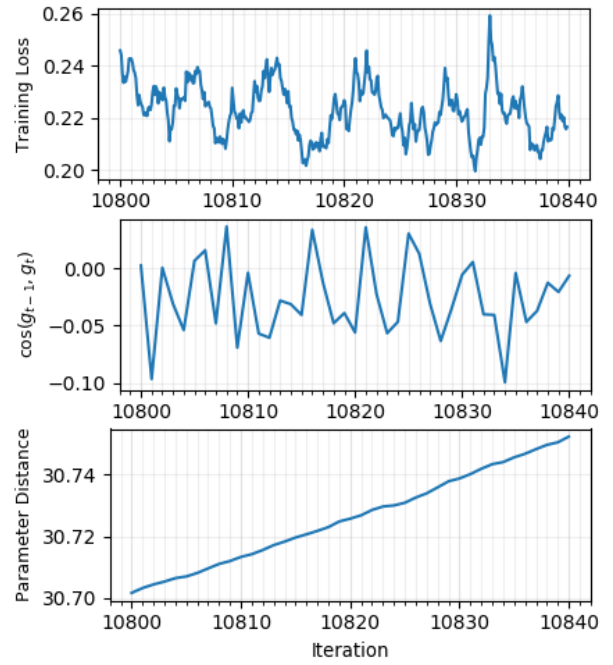
Figure 15. Plots for VGG-11 Epoch 25 trained using **SGD** on CIFAR-10. The descriptions of the plots are same as in figure 1.
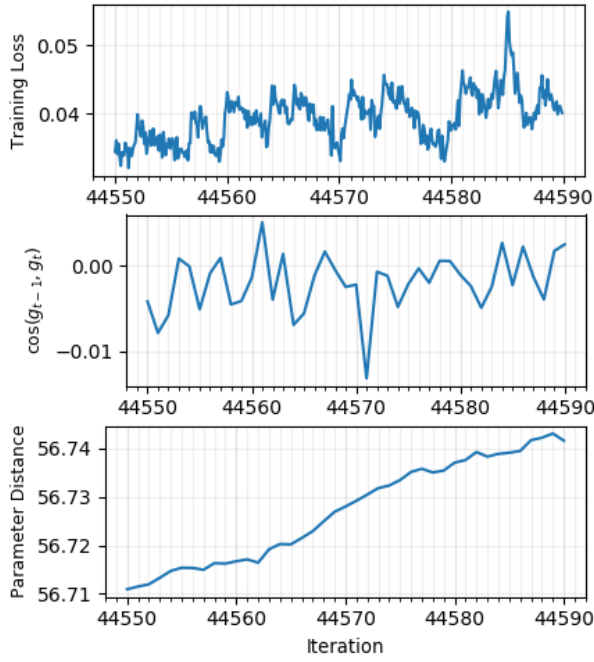
Figure 16. Plots for VGG-11 Epoch 100 trained using **SGD** on CIFAR-10. The descriptions of the plots are same as in figure 1.


Figure 18. Plots for MLP Epoch 1 trained using **SGD** on MNIST. The descriptions of the plots are same as in figure 1.

Figure 17. Plots for MLP Epoch 1 trained using full batch **Gradient Descent (GD)** on MNIST. The descriptions of the plots are same as in figure 1.

Figure 19. Plots for MLP Epoch 2 trained using **SGD** on MNIST. The descriptions of the plots are same as in figure 1.

Figure 20. Plots for VGG-11 Epoch 1 trained using full batch **Gradient Descent (GD)** on Tiny-ImageNet. The descriptions of the plots are same as in figure 1.

Figure 21. Plots for VGG-11 Epoch 1 trained using **SGD** on Tiny-ImageNet. The descriptions of the plots are same as in figure 1.
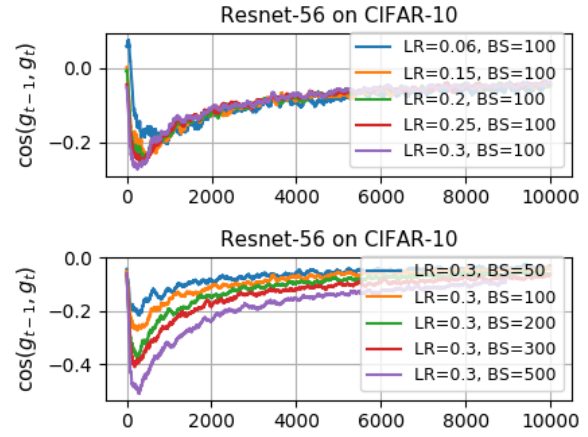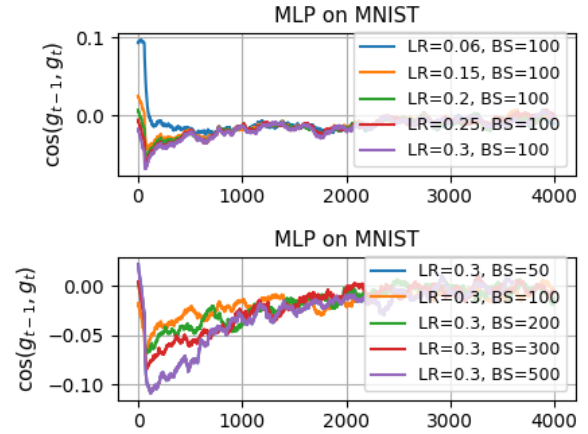

Figure 22. Changing batch size changes the cosine of angle between consecutive gradients while changing learning rate does not have any significant effect on the cosine. This shows batch size has a qualitatively different role compared with learning rate. Note that the curves are smoothened for visual clarity.
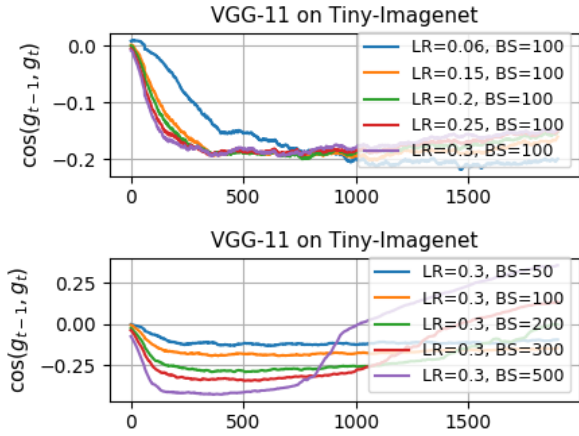

Figure 23. Changing batch size changes the cosine of angle between consecutive gradients while changing learning rate does not have any significant effect on the cosine. This shows batch size has a qualitatively different role compared with learning rate. Note that the curves are smoothened for visual clarity.

Figure 24. Changing batch size changes the cosine of angle between consecutive gradients while changing learning rate does not have any significant effect on the cosine. This shows batch size has a qualitatively different role compared with learning rate. Note that the curves are smoothened for visual clarity.
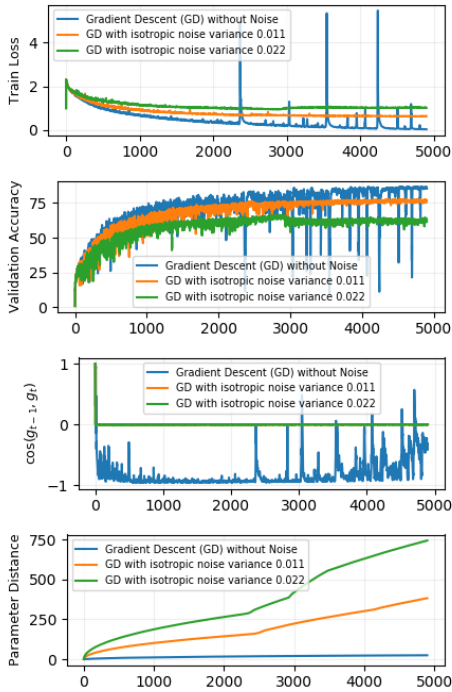

Figure 25. Plots for Resnet-56 trained by GD (without noise) and GD with artificial isotropic noise sampled from Gaussian distribution with different variances. Models trained using GD with added isotropic noise get stuck in terms of training loss and have worse validation performance compared with the model trained with GD.
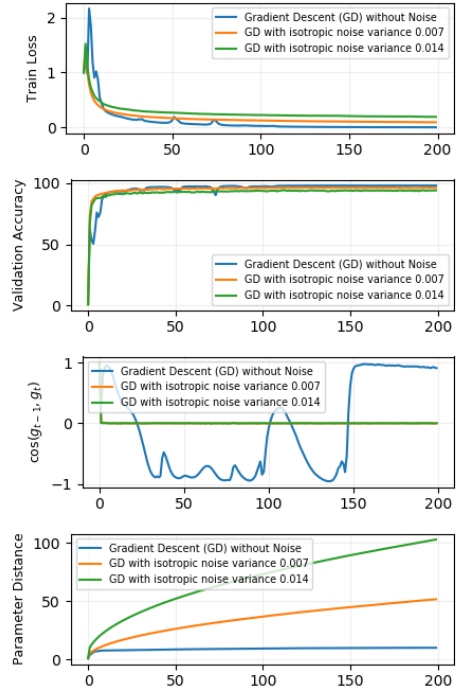

Figure 26. Plots for MLP trained by GD (without noise) and GD with artificial isotropic noise sampled from Gaussian distribution with different variances. Models trained using GD with added isotropic noise get stuck in terms of training loss and have worse validation performance compared with the model trained with GD.
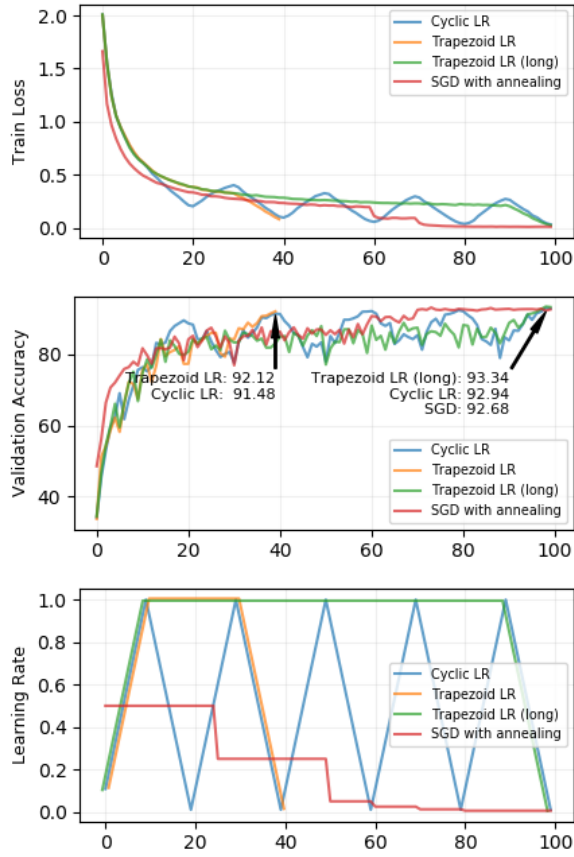
Figure 27. Plots for Resnet-56 on CIFAR-10 trained using Cyclic learning rate (CLR), SGD with stepsize annealing, and trapezoid schedule. Cycles in the CLR schedule are redundant, which is shown by the trapezoid schedule.
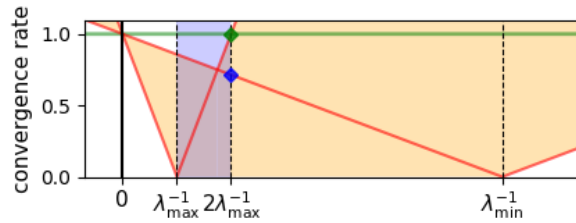

Figure 28. Graph of $|\lambda_i \eta - 1|$: convergence rates of gradient descent on $\lambda_{\min}$-strongly convex and $\lambda_{\max}$-smooth quadratic surfaces. The area which is shaded by orange contains possibly graphs of other eigenvalues of the hessian. In the range of learning rates shaded by blue, trajectory underdamps in the direction of the maximum eigenvalue. For a certain learning rate, while the trajectory oscillates in the direction of the maximum eigenvalue (green diamond), it overdamps in all the others (e.g. blue diamond - in a 'flat' direction).