Towards Flatter Loss Surface via Nonmonotonic Learning Rate Scheduling

Sihyeon Seong[1], **Yekang Lee**[1], **Youngwook Kee**[1], **Dongyoon Han**[1, 2], **Junmo Kim**[1]

[1]School of Electrical Engineering, KAIST
[2]Clova AI Research, NAVER Corp.
{sihyun0826,askhow,youngwook.kee,dyhan,junmo.kim}@kaist.ac.kr

Abstract

Whereas optimizing deep neural networks using stochastic gradient descent has shown great performances in practice, the rule for setting step size (i.e. learning rate) of gradient descent is not well studied. Although it appears that some intriguing learning rate rules such as ADAM (Kingma and Ba, 2014) have since been developed, they concentrated on improving convergence, not on improving generalization capabilities. Recently, the improved generalization property of the flat minima was revisited, and this research guides us towards promising solutions to many current optimization problems. In this paper, we analyze the flatness of loss surfaces through the lens of robustness to input perturbations and advocate that gradient descent should be guided to reach flatter region of loss surfaces to achieve generalization. Finally, we suggest a learning rate rule for escaping sharp regions of loss surfaces, and we demonstrate the capacity of our approach by performing numerous experiments.

1 INTRODUCTION

Overfitting is a core issue in the domain of machine learning. When it comes to deep learning, it becomes even more important, because of its high dimensionality. Owing to the huge number of parameters, deep learning models show some strange behaviors. For example, deep models easily fit random labeling of training data and furthermore training data replaced with random noise input (Zhang et al., 2016). A peculiar phenomenon called "fooling deep neural networks (DNNs)" was also reported in (Nguyen et al., 2015; Szegedy et al., 2013). These kinds of unexpected behavior might be a potential risk when adopting DNNs for applications which require high precision. Classic DNN regularization methods include placing a Gaussian or Laplacian prior on parameters, called weight decays (Figueiredo, 2003; Bishop, 2006). A weight decay assumes that the desirable solutions of the parameters are placed near zero, and therefore does not consider solutions which may lie a bit far from zero but possibly show better test performance. Numerous interesting works are still being proposed, including Stochastic Gradient Langevin Dynamics(SGLD) (Raginsky et al., 2017), parametrization method for reducing overfitting for genomics (Romero et al., 2016), employing stochastic effects; randomly dropping features (Hinton et al., 2012) or using stochastic depth (Huang et al., 2016). Some interesting works analyzed these stochastic effects in the view of a L_2-regularization (Wager et al., 2013) or an ensemble method (Singh et al., 2016).

The concept of generalization via achieving flat minima was first proposed in (Hochreiter et al., 1995), and its importance has recently been revisited in the domain of deep learning optimization (Chaudhari et al., 2016), (Keskar et al., 2016). This another viewpoint of thinking generalization may become a promising direction for investigating the weight space property of DNNs; moreover, DNN loss surface property analysis has become a popular issue (Sagun et al., 2016; Swirszcz et al., 2016; Littwin and Wolf, 2016; Im et al., 2016). Recently, achieving generalization via flat minima is investigated in terms of optimizing PAC-Bayes bound (Dziugaite and Roy, 2017a,b; Neyshabur et al., 2017).

A stochastic gradient descent (SGD) walks around loss surfaces in DNNs, and its behavior can be controlled by learning rates. It leads us to an interesting question: "Is it possible to seek flatter valleys of loss surfaces by controlling learning rate schedules?". We show the relation of learning rates and flatness of the loss surface, then show that a nonmonotonic scheduling of learning rates with an intermediate large learning rate stages are

beneficial to discover flat minima and therefore leads to improved generalization. On the scheduling of learning rates, recent studies achieve great convergence rates on training data, but they are prone to overfit, showing some degradation of test performance. Therefore, even state-of-the-art DNN models (Simonyan and Zisserman, 2014; Szegedy et al., 2014; He et al., 2016) have continued to use simple step or exponentially-decaying learning step sizes. Thus, our work has great implications on improving the theory of learning rates. To the best of our knowledge, this is the first work that pays substantial attention to learning rates with regard to generalization and overfitting of deep neural networks. We cite selected noteworthy work on learning rates; however, again note that none of the work concerns what we consider throughout the paper. Regarding fast convergence of given training data: RMSprop (Tieleman and Hinton, 2012), Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), Adam (Kingma and Ba, 2014), and an interesting work that searches for optimal adaptive learning rates without manual tuning (Schaul et al., 2012).

In this paper, we begin with robustness in the input space, which is a traditional way of explaining generalization. Then, we find the relation of robustness in the input perturbation and that in weight perturbation to show why flat minima work better (Section 2). Next, we explain how large learning rates can guide gradient descent to flatter losses (Section 3). Then, a simple nonmonotonic learning rate scheduling technique is introduced for adopting larger learning rates. Finally, our claims are demonstrated by performing numerous experiments.

2 THE RELATION OF ROBUSTNESS IN INPUT SPACE AND WEIGHT SPACE

In this section, we show that generalization can be achieved through robustness with respect to input perturbations, input perturbations can be equivalently transferred to weight perturbations, and therefore generalization can be achieved through robustness with respect to weight perturbations.

Generalization can be stated as the uniformity of the loss function with respect to input change (See Supplementary Material A). Denote $\mathbf{x} \in \mathbb{R}^{n\times 1}$ as input, $\mathbf{w} \in \mathbb{R}^d$ as the vectorized weight of the model and $\mathcal{L}(\mathbf{w};\mathbf{x})$ as the loss function of the neural network. Then, input perturbations $\boldsymbol{\delta}^{\mathbf{x}} \in \mathbb{R}^{n\times 1}$ should result in a small change in the loss function:

$$|\mathcal{L}(\mathbf{w};\mathbf{x}+\boldsymbol{\delta}^{\mathbf{x}}) - \mathcal{L}(\mathbf{w};\mathbf{x})| < \epsilon,$$

A similar interpretation of generalization in DNNs can be found in (Rifai et al., 2011), where the authors attempted to reduce $|f(\mathbf{w};\mathbf{x}+\boldsymbol{\delta}^{\mathbf{x}}) - f(\mathbf{w};\mathbf{x})|$ for f, which is an auto-encoder.

Now, the generalization capability of the model is evaluated for the change of the loss function with respect to perturbations on the weight vector $\boldsymbol{\delta}^{\mathbf{W}}$.

Lemma 1. *Let $\boldsymbol{\delta}^{\mathbf{x}}$ and $\boldsymbol{\delta}^{w}$ be input and weight perturbations, respectively. For a single-layer neural network, the input perturbations $\boldsymbol{\delta}^{\mathbf{x}}$ can be transferred to the weight perturbations $\boldsymbol{\delta}^{w}$. More formally, suppose we have weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ and weight perturbations $\boldsymbol{\delta}^{\mathbf{W}} \in \mathbb{R}^{m\times n}$. Then for any $\mathbf{W}$ and $\mathbf{x}$ that satisfy $\mathbf{x} \neq \mathbf{0}$ and $\mathbf{W} \neq \mathbf{0}$, there exists $\boldsymbol{\delta}^{\mathbf{W}}$ such that $(\mathbf{W}+\boldsymbol{\delta}^{\mathbf{W}})\mathbf{x} = \mathbf{W}(\mathbf{x}+\boldsymbol{\delta}^{\mathbf{x}})$. Consequently, $|\mathcal{L}(\mathbf{W}+\boldsymbol{\delta}^{\mathbf{W}};\mathbf{x}) - \mathcal{L}(\mathbf{W};\mathbf{x})| = |\mathcal{L}(\mathbf{W};\mathbf{x}+\boldsymbol{\delta}^{\mathbf{x}}) - \mathcal{L}(\mathbf{W};\mathbf{x})|$.*

Proof. The proof can be accomplished by finding $\boldsymbol{\delta}^{\mathbf{W}}$ which satisfies

$$\boldsymbol{\delta}^{\mathbf{W}}\mathbf{x} = \mathbf{W}\boldsymbol{\delta}^{\mathbf{x}} \tag{1}$$

The following choice of $\boldsymbol{\delta}^{\mathbf{W}}$ satisfies (1):

$$\boldsymbol{\delta}^{\mathbf{W}} = \frac{\mathbf{W}\boldsymbol{\delta}^{\mathbf{x}}}{\mathbf{x}^\top\mathbf{x}}\mathbf{x}^\top = \frac{\mathbf{W}\boldsymbol{\delta}^{\mathbf{x}}}{\|\mathbf{x}\|^2}\mathbf{x}^\top \tag{2}$$

□

Proposition 1. *Suppose $|\mathcal{L}(\mathbf{W}+\boldsymbol{\delta}^{\mathbf{W}};\mathbf{x}) - \mathcal{L}(\mathbf{W};\mathbf{x})| < \epsilon$ holds for any $\boldsymbol{\delta}^{\mathbf{W}}$ such that $\left\|\boldsymbol{\delta}^{\mathbf{W}}\right\|_F < \delta$, $\delta > 0$ and $\epsilon > 0$. Then $|\mathcal{L}(\mathbf{W};\mathbf{x}+\boldsymbol{\delta}^{\mathbf{x}}) - \mathcal{L}(\mathbf{W};\mathbf{x})| < \epsilon$ holds for any $\boldsymbol{\delta}^{\mathbf{x}}$ such that $\frac{\|\boldsymbol{\delta}^{x}\|}{\|\mathbf{x}\|} < \frac{\delta}{\sigma_{max}(\mathbf{W})}$, where $\sigma_{max}(\mathbf{W})$ is the maximum singular value of $\mathbf{W}$.*

Proof. For any $\boldsymbol{\delta}^{\mathbf{x}}$ such that $\frac{\|\boldsymbol{\delta}^{\mathbf{x}}\|}{\|\mathbf{x}\|} < \frac{\delta}{\sigma_{max}(\mathbf{W})}$, if we choose $\boldsymbol{\delta}^{\mathbf{W}}$ as in (2), then from the result of Lemma 1, we have

$$\begin{aligned}&|\mathcal{L}(\mathbf{W};\mathbf{x}+\boldsymbol{\delta}^{\mathbf{x}}) - \mathcal{L}(\mathbf{W};\mathbf{x})| \\ &\qquad = |\mathcal{L}(\mathbf{W}+\boldsymbol{\delta}^{\mathbf{W}};\mathbf{x}) - \mathcal{L}(\mathbf{W};\mathbf{x})|\end{aligned} \tag{3}$$

Then, corresponding bound on $\left\|\boldsymbol{\delta}^{\mathbf{W}}\right\|_F$ can be derived by

$$\left\|\boldsymbol{\delta}^{\mathbf{W}}\right\|_F^2 = \frac{\|\mathbf{W}\boldsymbol{\delta}^{\mathbf{x}}\|^2 \left\|\mathbf{x}^\top\right\|^2}{(\mathbf{x}^\top\mathbf{x})^2} \leq \frac{\sigma_{max}^2(\mathbf{W})\|\boldsymbol{\delta}^{x}\|^2}{\|\mathbf{x}\|^2} < \delta^2, \tag{4}$$

where we used $\left\|\mathbf{a}\mathbf{b}^\top\right\|_F^2 = \|\mathbf{a}\|^2\|\mathbf{b}\|^2$ and $\|\boldsymbol{W}\boldsymbol{\delta}^{x}\| \leq \sigma_{max}(\boldsymbol{W})\|\boldsymbol{\delta}^{\mathbf{x}}\|$.

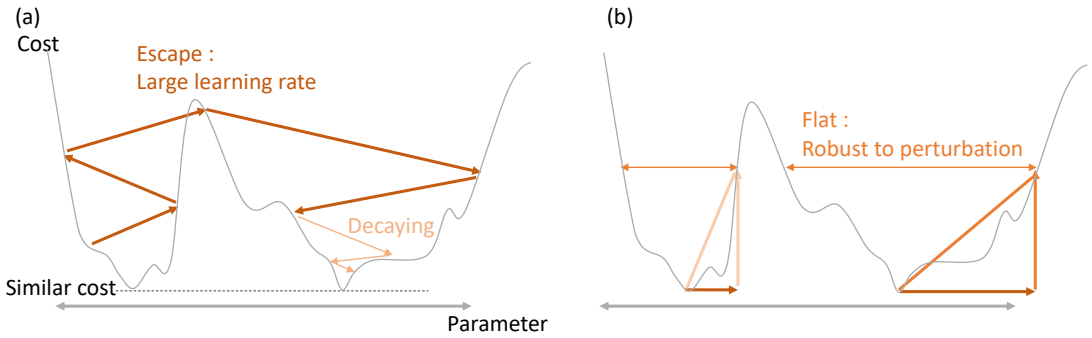


Figure 1: Conjectured illustration showing the relation of learning rate to loss surface. (a) A nonmonotonic scheduling of learning rates with an intermediate large learning rate stages lets the model escape from the steep valleys with high curvature. (b) Concept of what we call 'wide valleys'. The scale of perturbation should be large enough so that the loss surface can effectively cope with proper perturbations.

Therefore, we have

$$|\mathcal{L}(\mathbf{W};\mathbf{x}+\boldsymbol{\delta}^{\mathbf{x}}) - \mathcal{L}(\mathbf{W};\mathbf{x})| = |\mathcal{L}(\mathbf{W}+\boldsymbol{\delta}^{\mathbf{W}};\mathbf{x}) - \mathcal{L}(\mathbf{W};\mathbf{x})| < \epsilon \quad (5)$$

□

According to Lemma 1, for an arbitrary input perturbation, the first layer's weight matrix can be perturbed so that $\mathbf{W}(\mathbf{x}+\boldsymbol{\delta}^{\mathbf{x}}) = (\mathbf{W}+\boldsymbol{\delta}^{\mathbf{W}})\mathbf{x}$, i.e. the first layer's responses are the same for the two cases. Thus, the remaining layer's responses are also the same throughout, and the results in Lemma 1 and Proposition 1 are applicable to general DNNs. In Supplementary Material B, we provide additional analysis that shows how to distribute weight perturbations to all DNN layers to match the input perturbations.

Now that we can transfer the perturbations of input to those of weight, generalization can be interpreted on the loss surface. Reduction of the effects on the loss function with respect to perturbations on the input, requires us to reduce the effects with respect to perturbations on the weight vector. Finally, we propose the relation of the loss surface and a measure for generalization as follows:

Conjecture 1. *Given the same cost value, if the loss surface is flatter, then it is more likely for neural networks to be more generalized.*

3 THE RELATION OF LEARNING RATES AND GENERALIZATION

High-dimensional loss surfaces are regarded as non-convex and extremely difficult to visualize. Let us consider the SGD algorithm, which corresponds well to current large-scale problems. Not only are there abundant pathways that the SGD can follow, but the pathways are also highly dependent on learning rates. We can expect the loss surface of the model to have many locally convex areas and the learning rates to largely affect the outcome achieved by the algorithm. We justify this situation by evaluating the stationary characteristic of the stochastic gradients as the mean of the given values as follows:

$$\bar{\mathcal{L}}(\mathbf{w};\mathbf{x}) \coloneqq \mathbb{E}_{\mathbf{x}_i\sim\mathcal{D}}[\mathcal{L}(\mathbf{w};\mathbf{x}_i)], \quad (6)$$

where the training data $\mathbf{x}_i \in \mathbb{R}^{n\times 1}$ are drawn from a distribution $\mathcal{D}$. We can fit a quadratic approximation of the entire loss surface surrounding the local optimum $\mathbf{w}^*$ using the positive definite Hessian matrix

$$\bar{\mathcal{L}}(\mathbf{w};\mathbf{x}) \approx \bar{\mathcal{L}}(\mathbf{w}^*;\mathbf{x}) + \frac{1}{2}(\mathbf{w}-\mathbf{w}^*)^\top \mathbf{H}^*(\mathbf{w}-\mathbf{w}^*), \quad (7)$$

where $\mathbf{H}^*$ is the mean (over $\mathcal{D}$) of the Hessian of the loss function at $\mathbf{w}^*$. Let us denote γ_t as the learning rate at the iteration t. Then, the gradient descent can be calculated as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma_t \nabla\bar{\mathcal{L}}(\mathbf{w};\mathbf{x}) \quad (8)$$

$$\approx \mathbf{w}_t - \gamma_t \mathbf{H}^*(\mathbf{w}_t - \mathbf{w}^*). \quad (9)$$

$$\mathbf{w}_{t+1} - \mathbf{w}^* \approx (I - \gamma_t \mathbf{H}^*)(\mathbf{w}_t - \mathbf{w}^*). \quad (10)$$

If we assume small and smooth changes of the learning rate during some epochs (equivalently, $t_s \le t \le t_f$ where $|\gamma_t - \gamma_{t_s}| < \epsilon_\gamma$), providing constant γ_{t_s}, the weight vector can be measured by both the optimal and the initial points:

$$\mathbf{w}_t \approx \mathbf{w}^* + (I - \gamma_{t_s}\mathbf{H}^*)^{t-t_s}(\mathbf{w}_{t_s} - \mathbf{w}^*) \quad (11)$$

Then, we can strictly obtain the range of the learning rate capable of enforcing the convergence. If $\mathbf{H}^*$ is diagonal as (Schaul et al., 2012), and h_{max} is the maximum value

amongst the second-order gradients, the convergence criteria are expressed by the condition[1]

$$|1 - \gamma_{t_s} h_{max}| < 1, \tag{12}$$

which is equivalent to

$$0 < h_{max} < \frac{2}{\gamma_{t_s}}, \tag{13}$$

which provides the relationship of the learning rate and the curvature of the surface necessary to converge. A large learning rate has the critical requirement that the curvature of the surface should be sufficiently low to avoid diverging. Therefore, we anticipate that the SGD having a large learning rate allows for locating a smooth area.

Our remarks on the relationship between learning rates and generalization address "which local minimum should be chosen" as conceptually illustrated in Figure 1. Recent studies on loss surfaces, such as (Dauphin et al., 2014) and (Choromanska et al., 2015), both theoretically and empirically discovered that local minima are more likely to be located only where train losses are very close to those of the global minimum. Based on these results, we consider that the loss surface has basins of local minima that occur only at the bottom of the loss surface, i.e., $|\mathcal{L}(\mathbf{w}^*; \mathbf{x}) - \mathcal{L}_{gmin}| \approx \epsilon$ where $\mathcal{L}_{gmin}$ is the global minimum. When such a basin has a high curvature h_{max} violating (13), w keeps drifting from w^* because of (11) until it reaches another basin with a smaller curvature, a flatter region satisfying (13).

Additionally, once it reaches the entrance of a basin of smaller curvature, it is more likely to converge to a flatter local minimum. This is discussed in Section 6.3 implying that the average slope of the loss surface depends on the width of the basin's entrance. Finally, we propose the following chain of effects showing how generalization can be achieved via nonmonotonic learning rates scheduling:

High learning rate $\Rightarrow$ escape from high curvature valleys in weight space $\Rightarrow$ smooth region in weight space $\Rightarrow$ smooth region in input space $\Rightarrow$ improved generalization.

4 LEARNING RATES AND STOCHASTIC VARIANCE

The major claim of our work is that learning rates should be set large to escape sharp loss surface valleys. For setting large learning rates, curvature of loss surfaces is the only constraint in plain gradient descent cases. However, stochastic variance (Reddi et al., 2016) further interferes with convergence in SGD cases. Stochastic variance is an inherent variance of gradients caused by minibatch selection in SGD. When stochastic variance is large, learning rates should be set smaller to prevent divergence. Therefore, the maximum learning rate can be achieved when the stochastic variance is small.

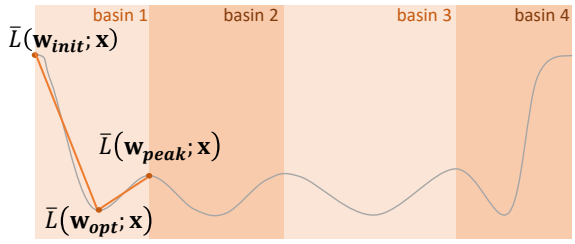


Figure 2: An illustration of the occurrence of maximum gradients. As the loss at random initial weights is significantly large, maximum gradients are most likely to occur near initial weights.

Stochastic variance is relatively large when the gradients are large (Johnson and Zhang, 2013). Therefore, learning rates should be set large after the occurrence of the maximum gradients. As in Figure 2, suppose the randomly initialized weights are $\mathbf{w}_{init}$ for which $\bar{\mathcal{L}}(\mathbf{w}_{init}; \mathbf{x})$ will be large (e.g. around $\log(n_{class})$ in the case of cross-entropy loss where n_{class} is the number of classes). $\mathbf{w}_{opt}$ is a local minimum that is closest to $\mathbf{w}_{init}$ and $\mathbf{w}_{peak}$ is a weights of local maximum which is the closest to $\mathbf{w}_{opt}$ and under the condition $\bar{\mathcal{L}}(\mathbf{w}_{init}; \mathbf{x}) \gg \bar{\mathcal{L}}(\mathbf{w}_{peak}; \mathbf{x})$. We constrain our claims under

$$\frac{|\bar{\mathcal{L}}(\mathbf{w}_{init}; \mathbf{x}) - \bar{\mathcal{L}}(\mathbf{w}_{opt}; \mathbf{x})|}{|\bar{\mathcal{L}}(\mathbf{w}_{peak}; \mathbf{x}) - \bar{\mathcal{L}}(\mathbf{w}_{opt}; \mathbf{x})|} \gg \frac{\|\mathbf{w}_{init} - \mathbf{w}_{opt}\|}{\|\mathbf{w}_{peak} - \mathbf{w}_{opt}\|} \tag{14}$$

and consider other cases to be beyond the scope of this paper. In practice, $\bar{\mathcal{L}}(\mathbf{w}_{init}; \mathbf{x}) \gg \bar{\mathcal{L}}(\mathbf{w}_{peak}; \mathbf{x})$. Therefore (14) is a probable condition. If we divide the loss surfaces into subspaces of basins and assume Lipschitz continuity, then we get the lower bound of maximum gradients during the optimization g_{LB} as follows.

$$g_{LB} = \frac{|\bar{\mathcal{L}}(\mathbf{w}_{init}; \mathbf{x}) - \bar{\mathcal{L}}(\mathbf{w}_{opt}; \mathbf{x})|}{\|\mathbf{w}_{init} - \mathbf{w}_{opt}\|} \tag{15}$$

which makes maximum gradients mostly occur near the initial weights.

[1](12) is still a valid condition for a non-diagonal $\mathbf{H}^*$ with the maximum eigenvalue h_{max}

5 PEAK LEARNING STAGE : NONMONOTONIC LEARNING RATES SCHEDULING

To experimentally show that high learning rates are beneficial to discovering flatter loss surfaces, we propose nonmonotonic learning rate schedules to utilize larger learning rates. Adopting larger learning rates presents the risk of learning to diverge because of (13) and Section 4. Rather than placing the maximum learning rate at the start of the learning, we place maximum learning rate in the middle, so the initial stage can stabilize learning(i.e. reduce stochastic variance) and prepare for the consecutive large learning rates. This learning rate schedule is motivated by the neurological phenomenon called *critical period* or *sensitive period*—a period in which learning plasticity reaches its peak (Wiesel et al., 1963; Ge et al., 2007). The plasticity gradually increases if biological systems are prepared to learn, after which plasticity is reduced over time. Considering the stabilization of learning, it is not surprising that plasticity *gradually* and *slowly* grows to a certain degree once it is switched on. We tried two nonmonotonic learning rate schedules, γ_t at normalized iteration t, as follows[2]:

- Gaussian Shape: $\gamma_t = \gamma_{max} * \exp(\frac{-(t-0.5)^2}{\sigma^2})$.
- Laplacian Shape: $\gamma_t = \gamma_{max} * \exp(-\frac{|t-0.5|}{\lambda})$.

Here, γ_{max} is the peak or maximum learning rate; γ_{min} is the final or minimum learning rate; and γ_{start} is the starting learning rate. Also, σ^2 for the Gaussian-shaped schedule and λ for the Laplacian-shaped schedule is defined as $-0.25 * \frac{1}{\ln(\gamma_{min}/\gamma_{max})}$ and $-0.5 * \frac{1}{\ln(\gamma_{min}/\gamma_{max})}$, respectively. Because the function starts with a value that is too small, we use a truncated function that is cut at the appropriate offset, t_{offset}, for faster convergence. The offset t_{offset} is $0.5 - \sqrt{-\sigma^2 \ln(\gamma_{start}/\gamma_{max})}$ in the Gaussian schedules and $0.5 + \lambda \ln(\gamma_{start}/\gamma_{max})$ in the Laplacian schedules.

The total number of iterations determines not only the total number of weight updates but also the sampling frequency from the continuous peak-shaped function. The sampling frequency controls the smoothness of the change in learning rate. By setting γ_{max} larger than the maximum learning rate of the classical learning rate schedule within convergence, the local optimum can be found in the flatter basin. We found both two peak shaped learning rates worked well, but the Gaussian-shaped learning rates worked slightly better than the Laplacian-shaped ones as shown in Figure 6. Therefore, we used Gaussian-shaped scheduling for most of the experiments. Note that a Gaussian-shaped curve is not the only way but just one way to implement the proposed peak learning stages.

[2]For notation simplicity, the iteration t is normalized between 0 and 1.

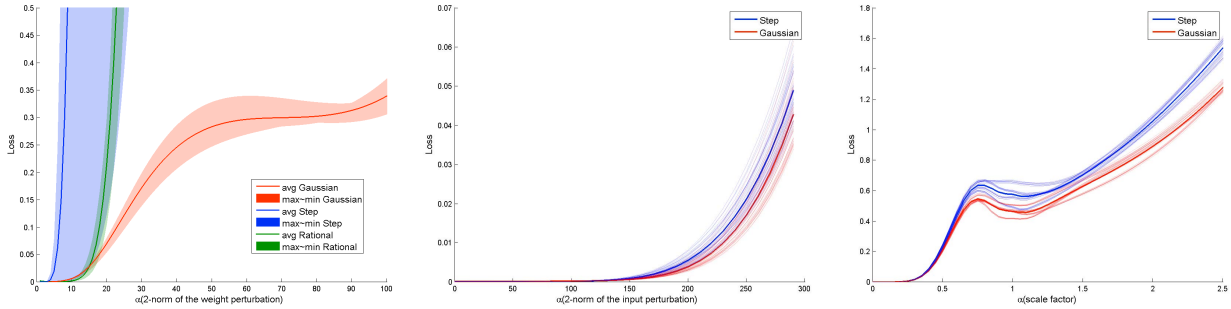
SLOW START Because of (13) and stochastic variance of the SGD, learning rates that are too large cause the learning to diverge. If we start with a too large learning rate, the learning either diverges or finds a poor critical point. Therefore, considering the stability of optimization, learning should commence with a small learning rate. However, we verified experimentally that starting with a learning rate that is too small does not lead to success. Thus, adopting t_{offset} in the learning rate rules is required to eliminate the redundant initial phase.

DECAYING LEARNING RATES The peak learning stage induces some divergence to escape from the sharp minima. Thus, the learning rate must be decayed for the final convergence. Considering conventional methods, this stage is intuitively acceptable. It is noteworthy that most DNNs are optimized using SGD, which implies loss surfaces are substantially fluctuating for each minibatch. Therefore decaying stages are necessary for achieving stability of optimization results.

6 EVALUATING THE FLATNESS OF LOSS SURFACES

Evaluation of the effect of the learning rate inevitably requires its behavior on the loss surface to be analyzed. However, in neural networks, the loss surface is high-dimensional so that optimization trajectory on the loss function becomes difficult to visualize. First, we tracked the local property of high-dimensional loss surfaces by measuring the magnitude of the gradient. Throughout this paper, the magnitude of the gradient refers to the 2-norm of the gradient vector which represents the local steepness of loss surfaces.

However, because the gradient (or Hessian) of the loss is computed at a specified location in the weight space, its application is limited to a local area of the loss surface. Therefore, we visualize large-scale properties of loss surfaces by adopting the linear path experiments introduced by (Goodfellow and Vinyals, 2014). The linear path experiments measure the loss surface by sweeping the trajectory between two points in high-dimensional spaces. By visualizing a single cross section of the loss surface, we indirectly measure its large-scale flatness.



(a) (b) (c)

Figure 3: Linear path experiments in Section 6. (a) Loss versus weight perturbation along the line path, which follows the trajectory between the final state weight and peak stage weight. The model trained using the proposed learning rate clearly shows flatter loss surface. Each color represents different learning rates. Maximum and minimum losses are also presented from different models trained by random initialization. (b) Loss versus randomly generated input perturbation. Each transparent line represents different random noise directions, and the bold line indicates the ensemble average of these transparent lines. The model trained using the proposed learning rate shows smaller loss for the same perturbation, indicating that our method is more robust to input perturbations. (c) Loss versus input perturbations in the directions of training to validation data. Each transparent line represents different perturbation decided by the selection of nearby validation data. The bold line indicates the ensemble average of the transparent lines. This also shows that the loss surface determined by our method is flatter with respect to input perturbations.

6.1 LOSS WITH RESPECT TO WEIGHT PERTURBATIONS

Because our learning rate rule has a peak stage that is not considered in existing methods, we analyzed its effects by performing linear path experiments which follow the trajectory between the final state weight, $\mathbf{w_{final}}$, and peak stage weight, $\mathbf{w}_{peak}$. That is, we calculated $\mathcal{L}(\mathbf{w};\mathbf{x})$, where $\mathbf{w} = \mathbf{w}_{final} + \alpha\frac{\mathbf{w}_{peak}-\mathbf{w}_{final}}{\|\mathbf{w}_{peak}-\mathbf{w}_{final}\|}$ and $\alpha > 0$. We reported the flatness over a training set S. Thus, the average of losses, $\frac{1}{|S|}\sum_{\mathbf{x}\in S}\mathcal{L}(\mathbf{w};\mathbf{x})$, was calculated. We also tested weight perturbations in random directions and fooling directions (Szegedy et al., 2013).

6.2 LOSS WITH RESPECT TO INPUT PERTURBATIONS

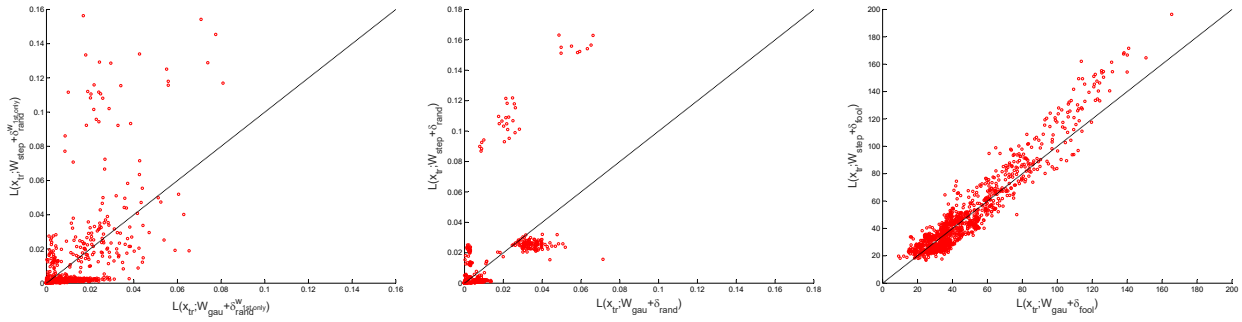
We also applied the linear path experiments to input spaces. Flatter loss surfaces can be easily expected to be robust, regarding random perturbations. Thus, we calculated $\mathcal{L}(\mathbf{w};\mathbf{x})$, where $\mathbf{x} = \mathbf{x}_{train} + \alpha\mathbf{x}_{noise}$, given that $\mathbf{x}_{noise}$ is randomly generated from the Normal distribution and then normalized to $\|\mathbf{x}_{noise}\| = 1$. Moreover, what the generalization tries to achieve is higher accuracy on the validation data. Therefore, we performed further experiments generating perturbations in the direction of validation data from training data. Thus, we calculated $\mathcal{L}(\mathbf{w};\mathbf{x})$, where $\mathbf{x} = \mathbf{x}_{train} + \alpha(\mathbf{x}_{val} - \mathbf{x}_{train})$. In this case, $\mathbf{x}_{val}$ was randomly selected from the ten closest validation candidates placed near $\mathbf{x}_{train}$ for each trial. Finally, the average of losses over training set is calculated.

6.3 THE FLATNESS OF LOSS SURFACES

The notion of flatness of a loss surface can be defined as follows. Let us consider a basin of a loss surface with local minimum $\mathbf{w}_i^*$, and define level set $\mathbf{S}_{\mathcal{L}_i} = \{\mathbf{w}|\mathcal{L}(\mathbf{w};\mathbf{x}) = c\}$ where c is the loss at the entrance of the basin. Choose any $\mathbf{w}_0 \in \mathbf{S}_{\mathcal{L}_i}$, and then the average slope of the surface along the line from $\mathbf{w}_i^*$ to $\mathbf{w}_0$ can be determined as $\frac{c-\epsilon}{\|\mathbf{w}_i^*-\mathbf{w}_0\|}$, where ϵ is the loss at $\mathbf{w}_i^*$. If this slope is small, then we call the loss surface is flat.

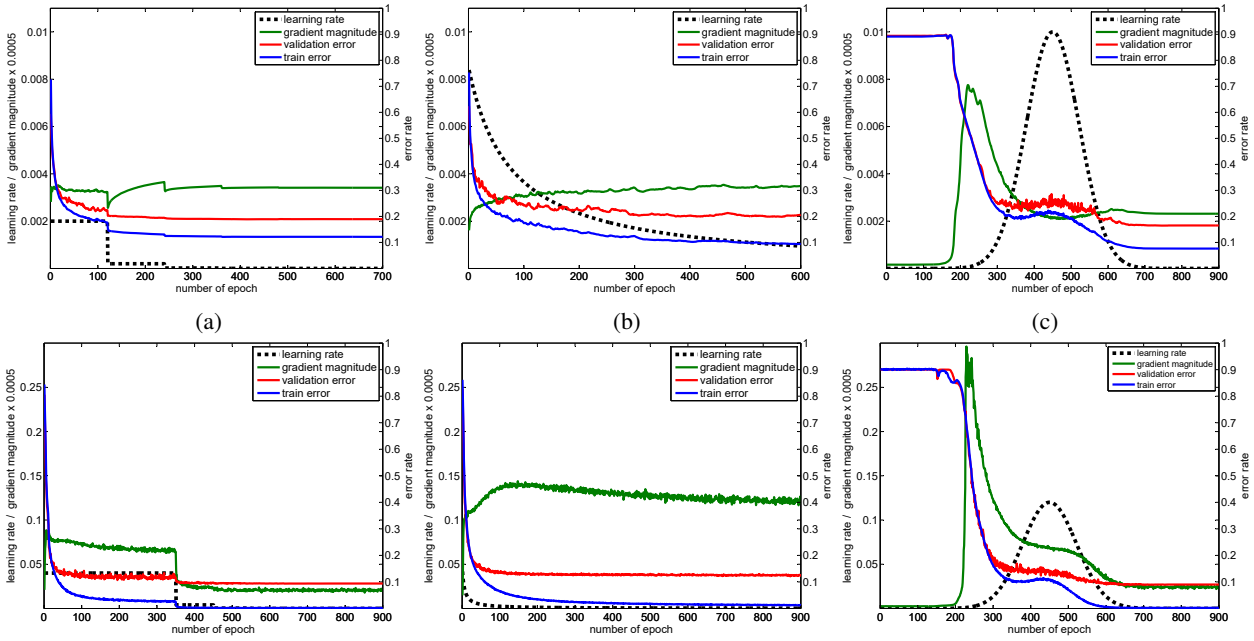
7 EXPERIMENTS

Here, we verify the roles of these peak-shaped learning rates on baseline convolutional neural networks (Krizhevsky et al., 2012), which we call "small model", and (Springenberg et al., 2014), which we call "large model" with the CIFAR-10 dataset. Hereinafter, all the experimental results are based on these settings, except for those in Section 7.2. We compared step decay, rational decay ($\gamma_t = \gamma_0(1+\lambda t)^{-1}$), and RMSprop (Tieleman and Hinton, 2012) as adaptive step size schedules and Gaussian-shaped schedules. For the "small model", the best validation error was 18.79 % with the baseline step decaying learning rates. This result was reduced to 16.48% after adding our learning rate method. For the "large model", the validation error, 9.53%, of the baseline method was reduced to 8.68%. We tried numerous random settings of hyperparameters (e.g., total number



(a) (b) (c)

Figure 4: Measuring effects of various weight perturbations (at $t = 1$) was introduced in Section 6. The x and y axes indicate the loss of the model trained by Gaussian and stepped learning rates, respectively. Because all graphs are tilted toward the y-axis, we can see that the nonmonotonic learning rate technique leads to flatter region of loss surfaces with less loss for weight perturbation. (a) Random weight perturbations are added to the first layer. (b) Random weight perturbations are added to the entire layers. (c) Weight perturbations along the fooling direction are added to the entire layers.



(a) (b) (c)

(d) (e) (f)

Figure 5: Examples of learning process of each learning rate schedule. (a), (b) and (c) :"small model" results, (d), (e) and (f) :"large model" results. (a), (b) and (c), or (d), (e) and (f) show learning rates, gradient magnitude, train and validation errors of each learning rate schedule (step, rational and Gaussian learning rates), respectively. All given values are calculated after each epoch. For the Gaussian learning case, the magnitude of the gradient peaks at the initial phase. Afterwards, it gradually decreases until the learning rate peaks. Other plots are provided for comparison.

of iterations, weight decay, momentum, and initial or peak learning rate) on the "small model" and confirmed that our methods are robust to selection of those settings (see Figure 7). However, we would like to clarify some experimental details on hyperparameters.

STARTING OFFSET This parameter can be ignored if the training time does not matter. It is safe to set $t_{offset} = 0$. However, going through all these extremely small learning rates is not necessary for achieving good performance. t_{offset} is controlled by γ_{start} and setting this value as maximum learning rate from the step decay-

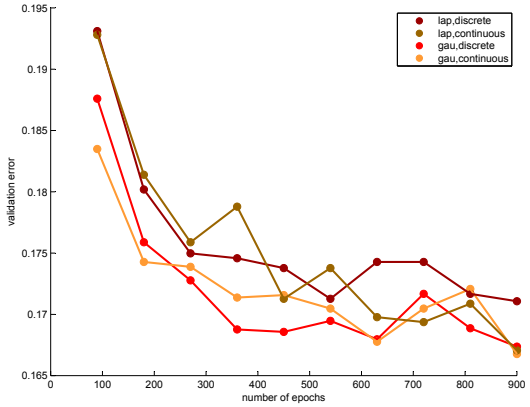


Figure 6: Comparisons of Gaussian-shaped and Laplacian-shaped learning rates, in terms of final validation error at $s = 1$. Horizontal axis is the total number of epochs. "Continuous" refers to learning rates that are updated at each iteration, whereas "Discrete" refers learning rates that are only updated at the beginning of each epoch.

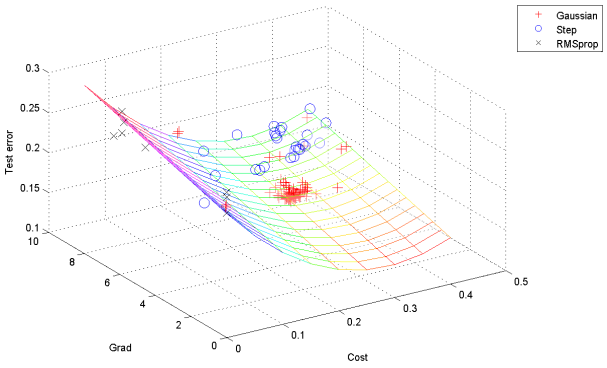


Figure 7: Prediction on test error with cost and ℓ_2-norm of gradients.

ing scheduling is predominantly a good choice.

MAXIMUM LEARNING RATE Because we are claiming that a large learning rate increases the generalization capability of the model, it is necessary to increase the peak learning rate more than three times of the original maximum learning rate reported in the baseline model. Three times the conventional maximum learning rate is most often safe. However, one can further increase γ_{max} if enough epochs are taken.

MINIMUM LEARNING RATE Generally, using learning rates that are too small presents the risk of overfitting. However, because we regularize for the smooth loss surface in the peak learning stages, setting γ_{min} smaller than the conventional value does not harm the performance. It sometimes results in further performance improvement during trials. We experimentally found that setting $\gamma_{min} = \gamma_{max}/10000$ works well.

THE NUMBER OF EPOCHS Our method spends times on peak learning rate, which does not reduce training loss. Therefore, we were required to take more epochs than the conventional schedule to get the best performance. However, even with the same number of epochs, our method can surpass other learning rate schedules, as reported in Section 7.2. Furthermore, we compared the performance of our method to the step-decaying learning rate using the model ensemble in Table 1. To reproduce the same performance as our method, the step decaying learning rate rule needs five model ensembles. Therefore, whereas Gaussian learning rate requires more epochs, our method is still practically efficient.

ROBUSTNESS TO HYPERPARAMETER TUNING We reported the experimental results with several values for the peak learning rate, γ_{max}, and the total number of iterations in Figure 7. This clearly shows our method is robust to variations in the hyperparameter. For a wide range of γ_{max}, from 0.002 to 0.015, the worst validation error was 18.15%, which is still better than the baseline error (18.79%).

7.1 EVALUATING FLATNESS OF THE LOSS GUIDED BY PEAK LEARNING STAGES

Using the techniques presented in Section 6, we evaluated the flatness of the loss trained by peak learning stages. Peak learning stage showed robustness with respect to random input perturbations (Figure 3(b)) and flatness in the direction of train to validation data (Figure 3(c)). In terms of weight perturbations, our method showed flatness in the direction of $\mathbf{w}_{final}$ to $\mathbf{w}_{peak}$ (Figure 3(a)). It also showed flatness along random directions (Figure 4(a), (b)) and even with fooling directions (Figure 4(c)). Thus, our theory of flattening loss surface using peak learning stages is well supported by numerous experimental results.

Method	Error (%)	
	Step learning rate	Gaussian learning rate
1 CNN	18.79(360 epochs)	16.91(360 epochs)
2 CNNs	17.49	15.98
3 CNNs	17.13	15.37
4 CNNs	17.09	14.88
5 CNNs	16.75	14.89

Table 1: Model Ensemble.

7.2 PERFORMANCE EVALUATION

In what follows, all test error rates are evaluated at the last epoch, not at the best validation epoch. The per-

formance of our learning rate schedule(Gaussian-shaped learning rates) is compared to the state-of-the-art models (Zeiler and Fergus, 2013; Lin et al., 2013; Goodfellow et al., 2013; Lee et al., 2014; He et al., 2016).

MNIST For MNIST, we set the $\gamma_{max} = 0.3$, $\sigma^2 = 2{*}0.08^2$ (this value makes Gaussian-shape visually looks good) of the Gaussian shaped-learning schedule with total 167 epochs. We considered that the training time is not the issue in case of MNIST because training time is significantly short. Throughout our proposed learning rate policy, without any modification of loss functions and architectures, the accuracy already surpasses the previous works. Table 2 summarizes the result.

METHOD	ERROR (%)
CNN	0.53
STOCHASTIC POOLING	0.47
NIN	0.47
MAXOUT NETWORKS	0.45
DEEPLY SUPERVISED NET	0.39
NIN + ours	**0.34**

Table 2: Test error rates for the MNIST dataset (without data augmentation).

CIFAR-10 and CIFAR-100 On the hyperparameter tuning, we followed same settings as in MNIST, except 720 total epochs for NIN (Lin et al., 2013), 270 epochs for ResNet (He et al., 2016). We preprocess the images similar to (Lin et al., 2013; Zeiler and Fergus, 2013; Goodfellow et al., 2013). Color jittering is also added in the ResNet experiment. For CIFAR-100, experiments on DenseNet (Huang et al., 2017) and Wide ResNet (Zagoruyko and Komodakis, 2016) are also performed. In case of Wide ResNet, we exceptionally set the peak learning rate twice as large as the start learning rate (this model adopts large dropout rate 0.3 and exceptionally small number of epochs, which interfere with convergence). Table 3 summarizes the result.

ImageNet Because ImageNet classification requires huge computational cost, we reduced shape of the Gaussian-shaped learning rate scheduling by adopting t_{offset}. We tested with the model of (Krizhevsky et al., 2012) and (Szegedy et al., 2014), which is well-reported and widely used as baseline model. First, we tried learning rate schedules with $t_{offset} = 0$, $\sigma^2 = 0.0128$ and total 270 epochs (Naive version). Then, to make training efficient, we adopt $\gamma_{start} = 0.01$, $\gamma_{max} = 0.03$ and $\gamma_{min} = 0.000003$, as suggested in the section 7. Finally, we report validation errors in Table 4.

METHOD	ERROR (%)	
	CIFAR-10	CIFAR-100
MAXOUT NETWORKS	9.38	-
DROPCONNECT	9.32	-
NIN	8.81	-
DEEPLY SUPERVISED NET	8.22	-
NIN + APL UNITS	7.51	-
RESNET(110-DEPTH)	6.41±0.21	27.815±0.15
DENSENET-BC (L=100, K=12)	-	22.47
DENSENET-BC (L=190, K=40)	-	17.18
WIDE RESNET (WRN-28-10-DROPOUT)	-	18.44
NIN + ours	**7.22**	-
ResNet(110-depth) + ours	**5.34±0.11**	**25.71±0.07**
DenseNet-BC (L=100, k=12) + ours	-	**22.17**
DenseNet-BC (L=190, k=40) + ours	-	**16.86**
Wide ResNet (WRN-28-10-dropout) + ours	-	**18.03**

Table 3: Test error rates for CIFAR-10 and CIFAR-100.

METHOD	ERROR (%)
ALEXNET(90 EPOCHS)	19.81
GOOGLENET(80 EPOCHS)	10.82
BATCHNORMALIZATION (80 EPOCHS)	9.05
AlexNet + ours(90 epochs)	**19.67**
AlexNet + ours(180 epochs)	**19.06**
AlexNet + ours(Naive version, 270 epochs)	**18.89**
GoogLeNet + ours(80 epochs)	**10.39**
BatchNormalization + ours(80 epochs)	**8.68**

Table 4: Validation error rates for the ImageNet dataset.

8 CONCLUSION

We showed why a flatter loss generalizes better in the view of robustness. Then we presented the relationship of flat losses and learning rates. Inspired by neuroscience, we further proposed peak learning stages for improving high-dimensional DNNs. We thoroughly analyzed how such learning rates affect conventional deep networks. To the best of our knowledge, the work presented in this paper is the first work in this line of research bridging the gap between the learning rate scheduling and the regularization theory of deep learning.

ACKNOWLEDGEMENT

This work was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIP) (No. CRC-15-05-ETRI).

References

Bishop, C. M. (2006). Pattern recognition. *Machine Learning*.

Chaudhari, P., Choromanska, A., Soatto, S., and LeCun, Y. (2016). Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *AISTATS*.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

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

Dziugaite, G. K. and Roy, D. M. (2017b). Entropy-sgd optimizes the prior of a pac-bayes bound: Data-dependent pac-bayes priors via differential privacy. *arXiv preprint arXiv:1712.09376*.

Figueiredo, M. A. (2003). Adaptive sparseness for supervised learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1150–1159.

Ge, S., Yang, C.-h., Hsu, K.-s., Ming, G.-l., and Song, H. (2007). A critical period for enhanced synaptic plasticity in newly generated neurons of the adult brain. *Neuron*, 54(4):559–566.

Goodfellow, I. J. and Vinyals, O. (2014). Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, S., Schmidhuber, J., et al. (1995). Simplifying neural nets by discovering flat minima. *Advances in Neural Information Processing Systems*, pages 529–536.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. (2016). Deep networks with stochastic depth. *arXiv preprint arXiv:1603.09382*.

Im, D. J., Tao, M., and Branson, K. (2016). An empirical analysis of deep network loss surfaces. *arXiv preprint arXiv:1612.04010*.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. (2014). Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Littwin, E. and Wolf, L. (2016). The loss surface of residual networks: Ensembles and the role of batch normalization. *arXiv preprint arXiv:1611.02525*.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5949–5958.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436.

Raginsky, M., Rakhlin, A., and Telgarsky, M. (2017). Nonconvex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. (2016). Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840.

Romero, A., Carrier, P. L., Erraqabi, A., Sylvain, T., Auvolat, A., Dejoie, E., Legault, M.-A., Dubé, M.-P., Hussin, J. G., and Bengio, Y. (2016). Diet networks: Thin parameters for fat genomic. *arXiv preprint arXiv:1611.09340*.

Sagun, L., Bottou, L., and LeCun, Y. (2016). Singularity of the hessian in deep learning. *arXiv preprint arXiv:1611.07476*.

Schaul, T., Zhang, S., and LeCun, Y. (2012). No more pesky learning rates. *arXiv preprint arXiv:1206.1106*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, S., Hoiem, D., and Forsyth, D. (2016). Swapout: Learning an ensemble of deep architectures. *arXiv preprint arXiv:1605.06465*.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Swirszcz, G., Czarnecki, W. M., and Pascanu, R. (2016). Local minima in training of deep networks. *arXiv preprint arXiv:1611.06310*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4.

Wager, S., Wang, S., and Liang, P. S. (2013). Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359.

Wiesel, T. N., Hubel, D. H., et al. (1963). Single-cell responses in striate cortex of kittens deprived of vision in one eye. *J Neurophysiol*, 26(6):1003–1017.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

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

Zeiler, M. D. and Fergus, R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.