UNDERSTANDING WHY NEURAL NETWORKS GENERALIZE WELL THROUGH GSNR OF PARAMETERS

Anonymous authors
Paper under double-blind review

ABSTRACT

As deep neural networks (DNNs) achieve tremendous success across many application domains, researchers tried to explore in many aspects on why they generalize well. In this paper, we provide a novel perspective on these issues using the gradient signal to noise ratio (GSNR) of parameters during training process of DNNs. The GSNR of a parameter is simply defined as the ratio between its gradient's squared mean and variance, over the data distribution. Based on several approximations, we establish a quantitative relationship between model parameters' GSNR and the generalization gap. This relationship indicates that larger GSNR during training process leads to better generalization performance. Futher, we show that, different from that of shallow models (e.g. logistic regression, support vector machines), the gradient descent optimization dynamics of DNNs naturally produces large GSNR during training, which is probably the key to DNNs' remarkable generalization ability.

1 INTRODUCTION

Deep neural networks typically contain far more trainable parameters than training samples, which seems to easily cause a poor generalization performance. However, in fact they usually exhibit remarkably small generalization gaps. Traditional generalization theory such as VC dimension (Vapnik & Chervonenkis, 1991) or Rademacher complexity (Bartlett P L, 2002) cannot explain its mechanism. Extensive research focuses on the generalization ability of DNNs (Neyshabur et al., 2017; S. et al., 2018; Keskar et al., 2016; Dinh et al., 2017; Hoffer et al., 2017; R et al., 2018; Dziugaite & Roy, 2017; etc, 2018; Kawaguchi et al., 2017; Advani & Saxe, 2017).

Unlike that of shallow models such as logistic regression or support vector machines, the global minimum of high-dimensional and non-convex DNNs cannot be found analytically, and can only be approximated by gradient descent and its variants (Zeiler, 2012; Kingma & Ba, 2014; Graves, 2013). Previous work (Zhang et al., 2016; Hardt et al., 2015; Dziugaite & Roy, 2017) suggests that the generalization ability of DNNs is closely related to gradient descent optimization. For example, Hardt et al. (2015) claims that any model trained with stochastic gradient descent (SGD) for reasonable epochs would exhibit small generalization error. Their analysis is based on the smoothness of loss function. In this work, we attempt to understand the generalization behavior of DNNs through GSNR and reveal how GSNR affects the training dynamics of gradient descent.

The GSNR of a parameter is defined as the ratio between its gradient's squared mean and variance over the data distribution. Previous work tried to use GSNR to conduct theoretical analysis on deep learning. For example, Rainforth et al. (2018) used GSNR to analyze variational bounds in unsupervised DNNs such as variational auto-encoder (VAE). Here we focus on analyzing the relation between GSNR and the generalization gap.

Intuitively, GSNR measures the similarity of a parameter's gradients among different training samples. Large GSNR implies that most training samples agree on the optimization direction of this parameter, thus the parameter is more likely to be associated with a meaningful "pattern" and we assume its update could lead to a better generalization. In this work, we prove that the GSNR is strongly related to the generalization performance, and larger GSNR means a better generalization.

To reveal the mechanism of DNNs' good generalization ability, we show that the gradient descent optimization dynamics of DNN naturally leads to large GSNR of model parameters, thus a good

generalization. Futher, we give a complete analysis and a detailed interpretation to this phenomenon. We believe this is probably the key to DNNs remarkable generalization ability.

In the remainder of this paper we first analyze the relation between GSNR and generalization (Section 2). We then show how the training dynamics lead to large GSNR of model parameters experimentally and analytically in Section 3.

2 LARGER GSNR LEADS TO BETTER GENERALIZATION

In this section, we establish a quantitative relation between the GSNR of model parameters and the generalization gap based on several proper approximations. This relationship indicates that larger GSNR during training process leads to better generalization performance.

2.1 GRADIENTS SIGNAL TO NOISE RATIO

Consider a data distribution $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ from which each sample $s = (x, y)$ is drawn, and a model $\hat{y} = f(x, \theta)$ trying to predict y from x, parameterized by θ. We use $\mathbf{g}_s(\theta)$ to denote parameters' gradient w.r.t sample s. Given the data distribution $\mathcal{Z}$, we can evaluate the (sample-wise) mean and variance of $\mathbf{g}_s(\theta)$. We denote them as $\mu(\theta) = \mathrm{E}_{s\sim\mathcal{Z}}(\mathbf{g}_s(\theta))$ and $\rho^2(\theta) = \mathrm{Var}_{s\sim\mathcal{Z}}(\mathbf{g}_s(\theta))$, respectively.

The gradient signal to noise ratio (GSNR) of q-th model parameter is defined as:

$$r_q(\theta) \equiv \frac{\mu_q^2(\theta)}{\rho_q^2(\theta)} \tag{1}$$

We can see, at a particular point of the parameter space, GSNR measures the similarity of the parameter gradients among different data samples.

2.2 ONE-STEP GENERALIZATION RATIO

In this section we introduce a new concept to help us measure the generalization performance during gradient descent optimization, which we call one-step generalization ratio (OSGR). We denote the training set as $D = \{(x_0, y_0), ..., (x_n, y_n)\} \sim \mathcal{Z}^n$ consists of n samples drawn from $\mathcal{Z}$, and the test set as $D' = \{(x'_0, y'_0), ..., (x'_{n'}, y'_{n'})\} \sim \mathcal{Z}^{n'}$. In practice we use the loss on D' to estimate the generalization loss. We denote the empirical training loss and test loss respectively as

$$L[D] = \frac{1}{n}\sum_{i=0}^{n} L(y_i, f(x_i, \theta)), \quad L[D'] = \frac{1}{n'}\sum_{i=0}^{n'} L(y'_i, f(x'_i)) \tag{2}$$

Then the empirical generalization gap is defined by $e = L[D'] - L[D]$.

In gradient descent optimization, both the training and test loss would decline step by step. We use $\Delta L[D]$ and $\Delta L[D']$ to denote the one-step training and test loss decrease during training, respectively. Let's consider the ratio between the expectations of $\Delta L[D']$ and $\Delta L[D]$ of one single training step, which we denote as $\mathcal{R}(\mathcal{Z})$.

$$\mathcal{R}(\mathcal{Z}) \equiv \frac{E_{D,D'\sim\mathcal{Z}^n}(\Delta L[D'])}{E_{D\sim\mathcal{Z}^n}(\Delta L[D])} \tag{3}$$

Note that the expectation of $\Delta L[D']$ is over D and D', and it's because the optimization step, aka training process, is on D. We refer $\mathcal{R}(\mathcal{Z})$ as OSGR of gradient descent optimization. Statistically the training loss decreases faster than the test loss and OSGR would be less than 1, which usually results in a non-zero generalization gap at the end of training. If OSGR is large in the whole training process, generalization gap will be small when training completes, implying good generalization ability of the model.

2.3 RELATION BETWEEN GSNR AND OSGR

In this section, we focus on deriving the relation between GSNR and OSGR, futher to prove the GSNR is strongly related to model generalization performance.

For simplicity, we assume the sizes of training and test datasets are equal, $n = n'$. In gradient descent optimization, we compute the mean of gradient over the training dataset, which we denote as $\mathbf{k}(\theta, D) = \frac{1}{n}\sum_{i=0}^{n}\mathbf{g}_i(\theta)$, and use it as the opposite descent direction. Note the difference between $\mu(\theta)$ and $\mathbf{k}(\theta, D)$, where $\mu(\theta)$ is the gradient mean over a data distribution and $\mathbf{k}(\theta, D)$ is the empirical gradient mean of a training dataset.

The gradient mean of the training and test datasets are, respectively

$$\mathbf{k}(\theta, D) = \frac{1}{n}\sum_{i=0}^{n}\mathbf{g}_i(\theta) = \frac{\partial L[D]}{\partial \theta} \quad , \quad \mathbf{k}'(\theta, D') = \frac{1}{n}\sum_{i=0}^{n}\mathbf{g}'_i(\theta) = \frac{\partial L[D']}{\partial \theta} \tag{4}$$

where $\mathbf{g}'_i(\theta)$ is the gradient w.r.t the i-th sample of the test dataset.

Both the training and test dataset are randomly generated from the same distribution $\mathcal{Z}^n$, so we can treat $\mathbf{k}(\theta, D)$ and $\mathbf{k}'(\theta, D')$ as random variables. At the beginning of the optimization process, θ is randomly initialized thus independent of D, so $\mathbf{k}(\theta, D)$ and $\mathbf{k}'(\theta, D')$ would obey the same distribution. After a period of training, the model parameters begin to fit the training dataset and become a function of D, $\theta = \theta(D)$, then the distribution of $\mathbf{k}(\theta(D), D)$ and $\mathbf{k}'(\theta(D), D')$ becomes different. However we will make our derivation under the non-overfitting limit approximation 2.3.1 stated as below.

Assumption 2.3.1 (Non-overfitting limit approximation) *In the early training stage, the mean of gradients w.r.t the training dataset and test dataset $\mathbf{k}(\theta(D), D)$ and $\mathbf{k}'(\theta(D), D')$ obey the same distribution, and we denote their mean and variance as $\mu(\theta)$ and $\sigma^2(\theta)$, i.e.*

$$\mathrm{E}_{D\sim\mathcal{Z}^n}[\mathbf{k}(\theta(D), D)] = \mathrm{E}_{D,D'\sim\mathcal{Z}^n}[\mathbf{k}'(\theta(D), D')] = \mu(\theta) \tag{5}$$

$$\mathrm{Var}_{D\sim\mathcal{Z}^n}[\mathbf{k}(\theta(D), D)] = \mathrm{Var}_{D,D'\sim\mathcal{Z}^n}[\mathbf{k}'(\theta(D), D')] = \sigma^2(\theta) \tag{6}$$

For simplicity, we denote $\mathbf{k}(\theta(D), D)$ and $\mathbf{k}(\theta(D), D')$ as $\mathbf{k}(\theta)$ and $\mathbf{k}'(\theta)$ respectively. We can get the following relation:

$$\mu(\theta) = \mathrm{E}_{D\sim\mathcal{Z}^n}[\mathbf{k}(\theta)] = \mathrm{E}_{D\sim\mathcal{Z}^n}[\frac{1}{n}\sum_{i=0}^{n}\mathbf{g}_i(\theta)] = \mathrm{E}_{s\sim\mathcal{Z}}[\mathbf{g}_s(\theta)] \tag{7}$$

$$\sigma^2(\theta) = \mathrm{Var}_{D\sim\mathcal{Z}^n}[\mathbf{k}(\theta)] = \mathrm{Var}_{D\sim\mathcal{Z}^n}[\frac{1}{n}\sum_{i=0}^{n}\mathbf{g}_i(\theta)] = \frac{1}{n}\mathrm{Var}_{s\sim\mathcal{Z}}[\mathbf{g}_s(\theta)] \equiv \frac{1}{n}\rho^2(\theta) \tag{8}$$

where $\sigma^2(\theta)$ is the variance of the gradient mean of a training dataset while $\rho^2(\theta)$ is the variance of the gradient of a single sample.

In one gradient descent step, the model parameter is updated by $\Delta\theta = \theta_{t+1} - \theta_t = -\lambda\mathbf{k}(\theta)$ where λ is the learning rate. If λ is small enough, the one-step training and test loss decrease can be approximated as

$$\Delta L[D] \approx -\Delta\theta \cdot \frac{\partial L[D]}{\partial \theta} + O(\lambda^2) = \lambda\mathbf{k}(\theta)\cdot\mathbf{k}(\theta) + O(\lambda^2) \tag{9}$$

$$\Delta L[D'] \approx -\Delta\theta \cdot \frac{\partial L[D']}{\partial \theta} + O(\lambda^2) = \lambda\mathbf{k}(\theta)\cdot\mathbf{k}'(\theta) + O(\lambda^2) \tag{10}$$

Usually there are some differences between the directions of $\mathbf{k}(\theta)$ and $\mathbf{k}'(\theta)$, so statistically $\Delta L[D]$ tends to be larger than $\Delta L[D']$ and the generalization gap would increase during training. When $\lambda \to 0$, in one single training step the empirical generalization gap increases by:

$$\Delta e \approx \lambda\mathbf{k}(\theta)\cdot\mathbf{k}(\theta) - \lambda\mathbf{k}(\theta)\cdot\mathbf{k}'(\theta) = \lambda(\mu(\theta)+\epsilon)(\mu(\theta)+\epsilon-\mu(\theta)-\epsilon') \tag{11}$$

$$= \lambda(\mu(\theta)+\epsilon)(\epsilon-\epsilon') \tag{12}$$

We replace the random variables, $\mathbf{k}(\theta) = \mu(\theta) + \epsilon$, $\mathbf{k}'(\theta) = \mu(\theta) + \epsilon'$. ϵ and ϵ' are random variables with zero mean and variance $\sigma^2(\theta)$. Since $E(\epsilon') = E(\epsilon) = 0$, ϵ and ϵ' are independent, the expectation of Δe is

$$E_{D,D'\sim\mathcal{Z}^n}(\Delta e) = E(\lambda\epsilon\cdot\epsilon) + O(\lambda^2) = \lambda\sum_{q=1}^{Q}\sigma_q^2 + O(\lambda^2) \tag{13}$$

where Q is the number of parameters and σ_q^2 is the variance of mean gradient of the q-th parameter. Consider the expectation of $\Delta L[D]$ and $\Delta L[D']$ when $\lambda \to 0$

$$E_{D\sim\mathcal{Z}^n}(\Delta L[D]) \approx \lambda E_{D\sim\mathcal{Z}^n}(\mathbf{k}(D)\cdot\mathbf{k}(D)) = \lambda\sum_{q=1}^{Q}E_{D\sim\mathcal{Z}^n}(k_q^2(D)) \tag{14}$$

$$E_{D,D'\sim\mathcal{Z}^n}(\Delta L[D']) = E_{D,D'\sim\mathcal{Z}^n}(\Delta L[D] - \Delta e) \approx \lambda\sum_{q=1}^{Q}(E_{D\sim\mathcal{Z}^n}(k_q^2(D)) - \sigma_q^2) \tag{15}$$

$$= \lambda\sum_{q=1}^{Q}(E_{D\sim\mathcal{Z}^n}(k_q^2(D)) - \rho_q^2/n) + O(\lambda^2) \tag{16}$$

Now we are ready to derive the relation between GSNR and OSGR, by substituting 16 and 14 into 3:

$$\mathcal{R}(\mathcal{Z}) = 1 - \frac{\sum_{q=1}^{Q}\rho_q^2}{n\sum_{q=1}^{Q}E_{D\sim\mathcal{Z}^n}(k_q^2(D))} \tag{17}$$

Now the right hand side of equation (17) can be estimated using only the samples in the training datasets. Because the gradient average $k_q(D)$ and sample-wise gradient variance ρ_q^2 can both be computed within one training dataset. We will elaborate on this estimation method in section 2.4.

Reformulate equation (17) as:

$$\mathcal{R}(\mathcal{Z}) = 1 - \frac{1}{n}\sum_{q=1}^{Q}\frac{E_{D\sim\mathcal{Z}^n}(k_q^2(D))}{\sum_{q'=1}^{Q}E_{D\sim\mathcal{Z}^n}(k_{q'}^2(D))}\frac{\rho_q^2}{E_{D\sim\mathcal{Z}^n}(k_q^2(D))} \tag{18}$$

$$= 1 - \frac{1}{n}\sum_{q=1}^{Q}\frac{E_{D\sim\mathcal{Z}^n}(k_q^2(D))}{\sum_{q'=1}^{Q}E_{D\sim\mathcal{Z}^n}(k_{q'}^2(D))}\frac{1}{r_q + \frac{1}{n}} \tag{19}$$

where $E_{D\sim\mathcal{Z}^n}(k_q^2(D)) = Var_{D\sim\mathcal{Z}^n}(k_q(D)) + E_{D\sim\mathcal{Z}^n}^2(k_q(D)) = \frac{1}{n}\rho_q^2 + \mu_q^2$.

We define $\Delta L_q[D]$ to be the training loss decrease caused by updating the q-th parameter. We can show that when λ is very small $\Delta L_q[D] = \lambda k_q^2(D) + O(\lambda^2)$. Therefore when $\lambda \to 0$

$$\mathcal{R}(\mathcal{Z}) = 1 - \frac{1}{n}\sum_{q=1}^{Q}W_q\frac{1}{r_q + \frac{1}{n}}, \quad \text{where } W_q = \frac{E_{D\sim\mathcal{Z}^n}(\Delta L_q[D])}{E_{D\sim\mathcal{Z}^n}(\Delta L[D])} \quad \text{with}\sum_{q=1}^{Q}W_q = 1 \tag{20}$$

Equation (20) shows that the GSNR (equation 1) plays a crucial role in the model's generalization ability. In the non-overfitting limit approximation and with a small enough learning rate, the one-step generalization ratio in gradient descent equals 1 minus the weighted average of $\frac{1}{r_q+\frac{1}{n}}$ over all model parameters divided by n. The weight is proportional to the expectation of the training loss decrease caused by updating that parameter. This result implies that larger GSNR of model parameters during training leads to smaller generalization gap growth and better generalization performance of the final model. Also note when $n \to \infty$, we have $\mathcal{R} \to 1$ and hence good generalization performance.

2.4 Experimental Verification of the relation between GSNR and OSGR

The relation between GSNR and OSGR, i.e. equation 17 or 20, is solid and can be accurately verified using any datasets if: (1) The dataset include enough number of samples to construct enough number

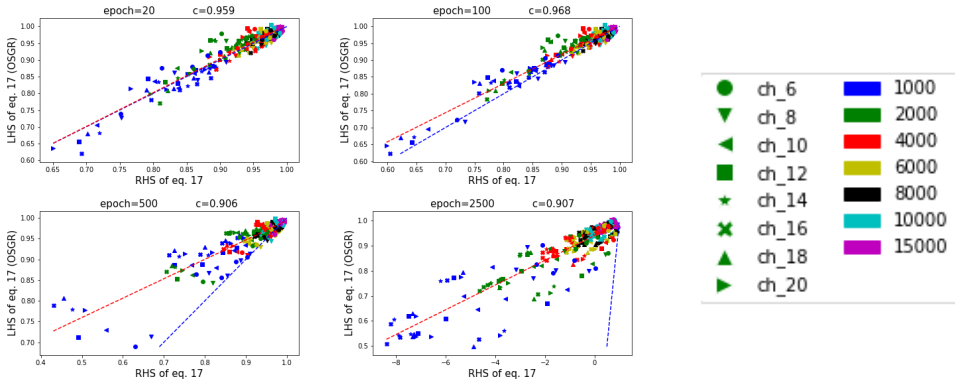

Figure 1: Left hand (LHS) and right side (RHS) of equation 17. Points are drawn under different experiment setting. Left figure: LHS vs RHS relation at epoch 20, 100, 500, 2500. Each point is drawn by LHS and RHS computed at the given epoch under different model structure (channel number) or training data size; red dotted line is the line of best fit computed by Least Square Method; blue dotted line is the line of reference representing LHS = RHS; the value of c in each title represents the Pearson correlation coefficient between LHS and RHS computed by points in figure. Right figure: The legend, different symbols and colors stand for different channel number and training data size respectively. Different random noise is not presented.

of training datasets and a large enough test dataset so that the estimation of ρ_q^2, $E_{D\sim\mathcal{Z}^n}(k_q^2(D))$ and OSGR can be accurate. (2) The learning rate is small enough. (3) In the early training stage of gradient descent.

To vefity equation 17, here we show how to estimate the left and right hand sides of it. Suppose we have M number of training datasets with n samples each and a large test dataset with n' samples. We initialize a model and train it separately on the M training datasets and test it with the same test dataset. For the t-th training iteration, we denote the training loss and test loss of the model trained on the m-th training dataset as L_{tm} and L'_{tm}, respectively. Then the left hand side, i.e. OSGR, of the t-th iteration can be estimated as

$$\mathcal{R}_t(\mathcal{Z}) \approx \frac{\sum_{m=1}^{M} L'_{t+1m} - L'_{tm}}{\sum_{m=1}^{M} L_{t+1m} - L_{tm}} \tag{21}$$

For the model trained on the m-th training datase, we can compute the t-th step gradient mean and sample-wise gradient variance of q-th parameter on the corresponding training dataset, denoted as $\hat{k}_{qmt}$ and $\hat{\rho}^2_{qmt}$, then the right hand side of equation 17 can be estimated

$$E_{D\sim\mathcal{Z}^n}(k_{qt}^2(D)) \approx \frac{1}{M}\sum_{m=1}^{M}\hat{k}^2_{qmt}, \quad \rho^2_{qt} \approx \frac{1}{M}\sum_{m=1}^{M}\hat{\rho}^2_{qmt} \tag{22}$$

Note that using equation 22, the right hand side can be computed merely with the samples in the training datasets.

To verify equation 17, we carry out the experiment on MNIST training dataset with simple CNNs which consists of 2 Conv-Relu-MaxPooling blocks and 2 fully-connnected layers. First, to estimate equation 22 with $M = 10$, we randomly sample 10 training datasets with n samples each and a test set with 10000 samples. Then to cover different conditions, we (1) set $n \in \{1000, 2000, 4000, 6000, 8000, 10000, 15000\}$, respectively; (2) inject noise in the datasets by randomly changing the labels with proportion $p_{random} \in \{0.0, 0.1, 0.2, 0.3, 0.5\}$; (3) experiment on various model structures which are different only at the number of channels in layers. For model details, please see Appendix A. Moreover, we use the gradient descent training (Not SGD), with a small learning rate of 0.001. The left and right hand sides of 17 at different epochs are shown in Figure 1, where each point represents one specific combination of above settings.

At the beginning of training, the data points closely distributed along the dashed line of $y = x$ which shows that equation 17 fits quite well under a variety of different settings. As training proceeds, equation 17 gradually losses its accuracy because the non-overfitting limit approximation no longer holds, but strong positive correlation between the left and right hand sides of equation 17 remains even when the training converges (at epoch 2500).

Through analytically derivations and experimental verifications, we prove that GSNR is strongly relatetd to OSGR which indicates the generalization ability, thus demonstrate that the larger GSNR during training leads to better generalization performance.

3 TRAINING DYNAMICS OF DNNS NATURALLY LEADS TO LARGE GSNR

In this section, we analyze and explain one observed phenomenon: the parameters' GSNR of DNN models rises in the early stages of training, while the GSNR of shallow models such as logistic regression or support vector machines declines during the entire training process. This difference of behaviors provides GSNR large practical values during DNN training, which in turn is associated with good generalization. We analyze the dynamics behind this phenomenon both experimentally and theoretically, which deepens our understanding of the good generalization ability of DNNs.

3.1 GSNR BEHAVIOR OF DNNS TRAINING

For shallow models, the GSNR of parameters decreases in the whole training process because gradients become small as learning proceeds when the optimizer finds the local minimum. But for DNN it is not the case. We trained DNNs on the CIFAR datasets and computed the GSNR averaged over all model parameters. Because $E_{D\sim\mathcal{Z}^n}(k_q^2(D)) = \frac{1}{n}\rho_q^2 + \mu_q^2$ and we assume n is large, $E_{D\sim\mathcal{Z}^n}(k_q^2(D)) \approx \mu_q^2$. In the case of only one large training datasets, we estimate GSNR of t-th iteration by

$$\hat{r}_{qt}(\theta) \approx \hat{k}_{qt}^2(\theta)/\hat{\rho}_{qt}^2(\theta) \tag{23}$$

As shown in Figure 2, the GSNR starts out low with randomly initialized parameters. As learning progresses, the GSNR increases in the early training stage and stays at a high level in the whole learning process. We also computed the proportion of the samples that have the same gradient sign (positive or negative) for each parameter, denoted as p_{same_sign}. In Figure 2c, we plot the mean timeseries of this proportion for all the parameters . This value increases from about 50%(half positive half negetive due to random initialization) at beginning to about 56% finally, which indicates that for most parameters, the gradient signs on different samples become to have a certain degree of consistency as the training proceeds. We hypothesis this is because meaningful features begin to emerge in the learning process and the gradients of the weights on these features tend to have the same sign among different samples.

Previous research (Zhang et al., 2016) showed that DNNs achieved zero training loss by memorizing training samples even if the labels were randomized. We also plot the average GSNR for model trained using data with randomized labels in Figure 2 and find that the GSNR stays at a low level throughout the training process. Although the training loss of both the original and randomized labels go to zero (not shown), the GSNR curves clearly distinguish between these two cases and reveal the lack of meaningful patterns in the latter one. We believe this is the reason why DNNs trained on real and random data lead to completely different generalization behavior.

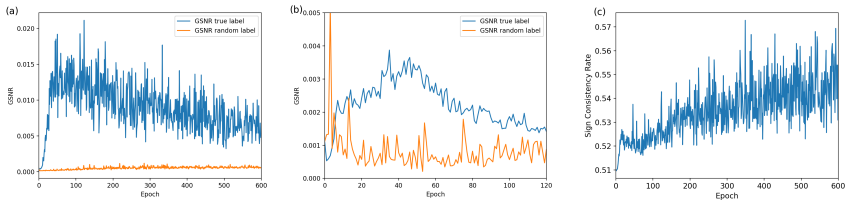

Figure 2: (a): GSNR curves generated by a simple network based on real and random data. An obvious upward process in the early training stage was observed for real data.(b): Same figure but for ResNet18 (c): average of $p_{same-sign}$ for the same model in (a)

3.2 TRAINING DYNAMICS BEHIND THE GSNR BEHAVIOR

In this section we show that the feature learning ability of DNNs is the key reason why the GSNR curve behavior of DNNs is different from that of shallow models during the gradient descent training process. To demonstrate this clearly, a simple two-layer perceptron regression model is constructed.

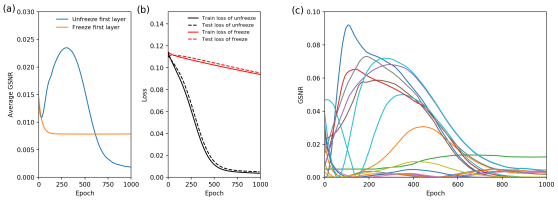

Figure 3: Average GSNR (a) and loss (b) curves for frozen and non-frozen the first layer. GSNR curves (c) of individual parameters for the non-frozen case

A synthetic dataset is generated as following. Each data point is constructed i.i.d. using $y = x_0 x_1 + \epsilon$, where x_0 and x_1 are drawn from uniform distribution $[-1, 1]$ and ϵ is drawn from uniform distribution $[-0.01, 0.01]$. The training set and test set sizes are 200 and 10000, respectively. We use a very simple 2-layer MLP structure with 2 input, 20 hidden neurons and 1 output.

We randomly initialized the model parameters and trained the model on the synthetic training dataset. As a control setup we also tried to freeze model weights in the first layer to prevent it from learning features. Note that a two layer MLP with the first layer frozen is equivalent to a linear regression model. That is, regression weights are learned on the second layer using fixed features extracted by the first layer. We plot the average GSNR of the second layer parameters for both the frozen and non-frozen cases. Figure 3 shows that in the non-frozen case, the average GSNR over parameters of the second layer has a significant upward process, whereas in the frozen case the average GSNR decreases in the beginning and remains at a low level during whole training process.

GSNR curve of individual parameters of the second layer of the non-frozen case are shown in Figure 3. It can be observed that the GSNR for some parameters have a significant upward process. We computed the Pearson correlation between the features extracted from the first layer and y corresponding to these parameters at the beginning of the training process and the maximum point of the GSNR curve. We can see that the learning process changes these features from a random initialized one to a "good" feature with stronger correlation with y as shown in Table 1. This shows that in the training of DNN, the feature learning process is closely related to the GSNR increasing process.

3.3 Analytical analysis of training dynamics behind DNNs' GSNR behavior

In this section, we will analytically investigate the training dynamics behind the GSNR curve behavior of DNNs. In the case of fully connected network structure, we can analytically show that the numerator of GSNR, the squared gradient mean of model parameters, tends to increase in the early training stage through feature learning.

In a fully connected network, whose parameters are denoted as $\theta \equiv \{\omega_1, \mathbf{b}_1, \omega_2, \mathbf{b}_2, ..., \omega_{l_{max}}, \mathbf{b}_{l_{max}}\}$, $\omega_1, \mathbf{b}_1$ is the weight and bias of the first layer, and so on. We denote the activations of the l-th layer as $[a_l]_{c_1}(\phi), c_1 = \{1, 2, ..., C_1\}$, where C_1 is the number of activations of this layer. ϕ is the collection of model parameters over all the layers before this layer, i.e. $\phi = \{\omega_1, \mathbf{b}_1, \omega_2, \mathbf{b}_2, ..., \omega_{l-1}, \mathbf{b}_{l-1}\}$. In the forward pass of the i-th sample, $\mathbf{a}_l(\phi)$ will be multiplied by the weight of this layer ω_l, which can be expressed as matrix multiplication.

$$[o_l]_{ic_2} = \sum_{c_1} [\omega_l]_{c_2 c_1} [a_l]_{ic_1}(\phi) \tag{24}$$

where $\mathbf{o}_l$ is the output of the matrix multiplication of the l-th layer and $c_2 = \{1, 2, ..., C_2\}$ where C_2 is the number of activations of the next layer. We use $\mathbf{k}_l$ to denote the gradient mean of weights of the l-th layer ω_l, i.e. $\mathbf{k}_l = \frac{1}{n} \sum_{i=1}^{n} \frac{L_i}{\partial \omega_l}$, where L_i is the loss of the i-th sample.

Here we analytically show that the feature learning ability of DNNs plays a crucial role in the GSNR increasing process. To be more precise, we show that the learning of features $\mathbf{a}_l(\phi)$, i.e. the learning of parameters ϕ, tends to increase the absolute value of $\mathbf{k}_l$. Let's consider, the one-step change of gradient mean $\Delta[k_l] = ([k_l])_{t+1} - ([k_l])_t$ with the learning rate $\lambda \to 0$. In one training step, θ will be updated by $\Delta\theta = \theta_{t+1} - \theta_t = -\lambda \mathbf{k}(\theta)$. Use linear approximation with $\lambda \to 0$, we have

$$\Delta[k_l]_{c_1 c_2} \approx \sum_{q=1}^{Q} \frac{\partial [k_l]_{c_1 c_2}}{\partial \theta_q} \Delta\theta_q = \sum_{q=1}^{Q_l} \frac{\partial [k_l]_{c_1 c_2}}{\partial \phi_q} \Delta\phi_q + \sum_{q=Q_l+1}^{Q} \frac{\partial [k_l]_{c_1 c_2}}{\partial \theta_q} \Delta\theta_q \tag{25}$$

where Q is the total number of model parameters and Q_l is the number of model parameters of all the layers before the l-th layer. We will focus on the first term of equation 25, i.e. the one-step

change of $\mathbf{k}_l$ caused by learning ϕ. Substituting $\mathbf{k}_l = \frac{1}{n}\sum_{i=1}^{n}\frac{L_i}{\partial\omega_l}$ and $\Delta\phi_q = (-\lambda\frac{1}{n}\sum_{i=1}^{n}\frac{\partial L_i}{\partial\phi_q})$ in to equation 25, we have

$$\Delta[k_l]_{c_1c_2} = -\frac{\lambda}{n^2}\sum_{q=1}^{Q_l}[\omega_l]_{c_1c_2}(\sum_{i=1}^{n}\frac{\partial L_i}{\partial[o_l]_{ic_2}}\frac{\partial[a_l]_{ic_1}}{\partial\phi_q})^2 + other\ terms \tag{26}$$

The detailed derivation of equation 26 can be found in Appendix B. We can see the first term (which is a summation of Q_l terms) in equation 26 has opposite sign with $[\omega_l]_{c_1c_2}$. This term will make $\Delta[k_l]_{c_1c_2}$ negatively correlated with $[\omega_l]_{c_1c_2}$. We plot the correlation between $\Delta[k_l]_{c_1c_2}$ with $[\omega_l]_{c_1c_2}$ for a model trained on MNIST for 200 epochs as shown in Figure 4a. In the early training stage, they are indeed negatively correlated. For top-10% weights with larger absolute values, the negative correlation is more significant.

Here we show that this negative correlation between $\Delta[k_l]_{c_1c_2}$ and $[\omega_l]_{c_1c_2}$ will tend to increase the absolute value of $[k_l]$ through an interesting mechanism. Consider the weights $[\omega_l]_{c_1c_2}$ with $\{[\omega_l]_{c_1c_2} > 0, [k_l]_{c_1c_2} < 0\}$. Learning ϕ would tend to decrease $[k_l]_{c_1c_2}$ and thus increase its absolute value because the first term in equation 26 is negative in this case. Learning $[\omega_l]_{c_1c_2}$ would increase $[\omega_l]_{c_1c_2}$ and its absolute value because $\Delta[\omega_l]_{c_1c_2} = -\lambda[k_l]_{c_1c_2}$ is positive in this case. This will form a positive feedback process, in which the numerator of GSNR, $([k_l]_{c_1c_2})^2$, would increase and so is the GSNR. Similar things happen for the weights with $\{[\omega_l]_{c_1c_2} < 0, [k_l]_{c_1c_2} > 0\}$.

Then what about the weights with $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} > 0\}$? Here we show that the weights with $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} > 0\}$ tends to change into weights with $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} < 0\}$ during training process. Consider the case where $\{[\omega_l]_{c_1c_2} > 0, [k_l]_{c_1c_2} > 0\}$, the first term in equation 26 will be negative, learning ϕ tends to decrease $[k_l]_{c_1c_2}$ or even change the sign of $[k_l]_{c_1c_2}$. Another posibility is that learning $[\omega_l]_{c_1c_2}$ tends to changes the sign of $[\omega_l]_{c_1c_2}$ because $\Delta[\omega_l]_{c_1c_2} = -\lambda[k_l]_{c_1c_2}$ is negative in this case. Both case will change the weights with $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} > 0\}$ to weights with $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} < 0\}$. Same things happen for the weights with $\{[\omega_l]_{c_1c_2} < 0, [k_l]_{c_1c_2} < 0\}$.

Therefore $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} < 0\}$ is a more stable state than $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} > 0\}$ in the early training process. For a simple model trained on Mnist, We plot the proportion of weights that satisfy $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} < 0\}$ in Figure 4b and find that there are more weights with $\{[\omega_l]_{c_1c_2}[k_l]_{c_1c_2} < 0\}$ than the opposite. Because weights with small absolute value easily change sign during training, we also plot this proportion for the top-10% weights with larger absolute values. It can be seen that for the weights with large absolute values, nearly 80% of these weights have opposite sign with their gradient mean as training proceeds. And for these weights, the numerator of GSNR, $([k_l]_{c_1c_2})^2$, tends to increase through a posotive feedback process in the early training stage of DNN.

Through discussions above, we further demonstrate that the learning of corresponding features tends to increase the GSNR of a weight based on analytical analysis. This GSNR increasing process leads to larger GSNR in the whole gradient descent training process of DNN, which in turn provides good generalization performance of DNN.

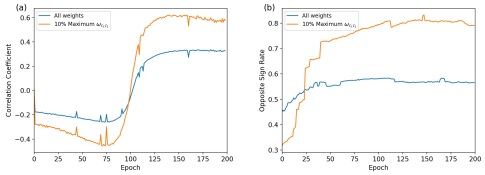

Figure 4: Dataset: Mnist. Left: Correlation between $\Delta[k_l]_{c_1c_2}$ with $[\omega_l]_{c_1c_2}$. Right : Ratio of weights that have opposite sign with its gradient mean.

Table 1: Pearson correlation of the features with y.

q	Beginning of training	Maximum of GSNR curve
1	-0.11	0.47
2	-0.33	0.53
3	-0.21	-0.27
4	0.07	0.40
5	0.11	0.44

4 SUMMARY

In this paper we investigated the relation between generalization of DNNs and the GSNR of the model parameters in gradient descent. We also analyzed the GSNR behavior and the mechanism behind it in the DNNs training process. Through our analysis, we hope to shed more light on the mechanisms behind DNNs impressive generalization ability.

REFERENCES

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

Mendelson S Bartlett P L. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1019–1028. JMLR. org, 2017.

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

Daniel Jakubovitz etc. Generalization error in deep learning, 2018.

Alex Graves. Agenerating sequences with recurrent neural networks, 2013. arXiv:1308.0850v5.

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

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

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

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

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

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Novak R, Bahri Y, Abolafia D A, Pennington J, and SohlDickstein J. Sensitivity and generalization in neural networks: An empirical study. *arXiv:1802.08760*, 2018.

Tom Rainforth, Adam R Kosiorek, Tuan Anh Le, Chris J Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. *arXiv preprint arXiv:1802.04537*, 2018.

Arora S., Ge R., Neyshabur B., and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach, 2018. arXiv:1802.05296.

Vladimir N Vapnik and A Ja Chervonenkis. The necessary and sufficient conditions for consistency of the method of empirical risk. *Pattern Recognition and Image Analysis*, 1(3):284–305, 1991.

Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012. arXiv:1212.5701.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A APPENDIX A

Model Structure in Section 2.4. As shown in Table 2, all models in the experiment consist of 2 Conv-Relu-MaxPooling blocks and 2 fully-connnected layers, but they are different at the number of channels. We choose the number of channel p as $p \in \{6, 8, 10, 12, 14, 16, 18, 20\}$.

Table 2: Model structure in Section 2.4. p is the number of channel in experiment setting and $q = int(2.5 * p)$

Layer	input channel number	output channel number
conv + relu + maxpooling	1	p
conv + relu + maxpooling	p	q
flatten	-	-
fc + relu	16 * q	10 * q
fc + relu	10 * q	10
softmax	-	-

B APPENDIX B

The step-by-step derivation of equation 26

$$\Delta[k_l]_{c_1c_2} = \sum_{q=1}^{Q_l} \frac{\partial [k_l]_{c_1c_2}}{\partial \phi_q} \Delta\phi_q + other\ terms \tag{27}$$

$$= \sum_{q=1}^{Q_l} \frac{\partial (\frac{1}{n}\sum_{i=1}^{n} \frac{\partial L_i}{\partial [\omega_l]_{c_1c_2}})}{\partial \phi_q} (-\lambda \frac{1}{n} \sum_{i=1}^{n} \frac{\partial L_i}{\partial \phi_q}) + other\ terms \tag{28}$$

$$= \sum_{q=1}^{Q_l} \frac{\partial (\frac{1}{n}\sum_{i=1}^{n} \frac{\partial L_i}{\partial [o_l]_{ic_2}} \frac{\partial [o_l]_{ic_2}}{\partial [\omega_l]_{c_1c_2}})}{\partial \phi_q} (-\frac{\lambda}{n} \sum_{i=1}^{n} \sum_{c_1'c_2'} \frac{\partial L_i}{\partial [o_l]_{ic_2'}} \frac{\partial [o_l]_{ic_2'}}{\partial [a_l]_{ic_1'}} \frac{\partial [a_l]_{ic_1'}}{\partial \phi_q}) + other\ terms \tag{29}$$

$$= -\frac{\lambda}{n^2} \sum_{q=1}^{Q_l} \frac{\partial (\sum_{i=1}^{n} \frac{\partial L_i}{\partial [o_l]_{ic_2}} [a_l]_{ic_1})}{\partial \phi_q} (\sum_{i=1}^{n} \sum_{c_1'c_2'} \frac{\partial L_i}{\partial [o_l]_{ic_2'}} [\omega_l]_{c_1'c_2'} \frac{\partial [a_l]_{c_1'}}{\partial \phi_q}) + other\ terms \tag{30}$$

$$= -\frac{\lambda}{n^2} \sum_{q=1}^{Q_l} \sum_{i=1}^{n} (\frac{\partial L_i}{\partial [o_l]_{ic_2}} \frac{\partial [a_l]_{ic_1}}{\partial \phi_q} + \frac{\partial^2 L_i}{\partial [o_l]_{ic_2} \partial \phi_q} [a_l]_{ic_1}) (\sum_{c_1'c_2'} [\omega_l]_{c_1'c_2'} \sum_{i=1}^{n} \frac{\partial L_i}{\partial [o_l]_{ic_2'}} \frac{\partial [a_l]_{c_1'}}{\partial \phi_q})$$
$$+ other\ terms \tag{31}$$

In the above derivation, we use relation $\frac{\partial [o_l]_{ic_2'}}{\partial [a_l]_{ic_1'}} = [\omega_l]_{c_1'c_2'}$ and $\frac{\partial [o_l]_{ic_2}}{\partial [\omega_l]_{c_1c_2}} = [a_l]_{ic_1}$ which can both be derived from equation 24. Consider the first term of equation 31 and when $c_1' = c_1, c_2' = c_2$ in the c_1', c_2' summation, we have

$$\Delta[k_l]_{c_1c_2} = -\frac{\lambda}{n^2} \sum_{q=1}^{Q_l} [\omega_l]_{c_1c_2} (\sum_{i=1}^{n} \frac{\partial L_i}{\partial [o_l]_{ic_2}} \frac{\partial [a_l]_{ic_1}}{\partial \phi_q})^2 + other\ terms \tag{32}$$

Note that the term related to $\frac{\partial^2 L_i}{\partial [o_l]_{ic_2} \partial \phi_q} [a_l]_{ic_1}$ and the terms when $c_1' \neq c_1$ or $c_2' \neq c_2$ in equation 31 is added into $other\ terms$ of equation 32.