Dropout in training neural networks: flatness of solution and noise structure

Zhongwang Zhang, Hanxu Zhou & Zhi-Qin John Xu [*]
School of Mathematical Sciences, Institute of Natural Sciences, MOE-LSC and
Qing Yuan Research Institute, Shanghai Jiao Tong University
`{0123zzw666, zhouhanxu, xuzhiqin}@sjtu.edu.cn`

Abstract

It is important to understand how the popular regularization method dropout helps the neural network training find a good generalization solution. In this work, we show that the training with dropout finds the neural network with a flatter minimum compared with standard gradient descent training. We further find that the variance of a noise induced by the dropout is larger at the sharper direction of the loss landscape and the Hessian of the loss landscape at the found minima aligns with the noise covariance matrix by experiments on various datasets, i.e., MNIST, CIFAR-10, CIFAR-100 and Multi30k, and various structures, i.e., fully-connected networks, large residual convolutional networks and transformer. For networks with piece-wise linear activation function and the dropout is only at the last hidden layer, we then theoretically derive the Hessian and the covariance of dropout randomness, where these two quantities are very similar. This similarity may be a key reason accounting for the goodness of dropout.

1 Introduction

Dropout is used with gradient-descent-based algorithms for training DNNs (Hinton et al., 2012; Srivastava et al., 2014), which drives the state-of-the-art test performance in deep learning (Tan and Le, 2019; Helmbold and Long, 2015). During training, the output of each neuron is multiplied with a random variable with probability p as $1/p$ and $1-p$ as zero. Note that p is called dropout rate, and every time for computing concerned quantity, the variable is randomly sampled at each feedforward operation. Dropout has been an indispensable trick in the training of deep neural networks (DNNs).

The noise structure in the training dynamics is important. For example, the noise structure of SGD helps find a flat solution (Keskar et al., 2016; Feng and Tu, 2021; Zhu et al., 2018). Similar to SGD, training with dropout is equivalent to that with some specific noise. The implicit regularization behind this specific noise structure finds solutions with better generalization (Hinton et al., 2012; Srivastava et al., 2014; Wei et al., 2020).

To understand what kind of noise benefits the generalization of training, in this work, we first study the characteristic of the minima found with the dropout regularization. We show that compared with the standard gradient descent (GD), the GD with dropout selects flatter minima. As suggested by many existing works (Keskar et al., 2016; Neyshabur et al., 2017; Zhu et al., 2018), flatter minima are more likely to have better generalization and stability.

To explain why dropout can find flat minima, we then explore the relation between the flatness of the loss landscape and the noise structure induced by dropout at minima through three methods and obtain consistent results as follows: i) Inverse variance-flatness relation: The noise is larger at the

---

sharper direction of the loss landscape; ii) Hessian-variance alignment relation: The Hessian of the loss landscape at the found minima aligns with the noise covariance matrix.

These two relations are intuitively consistent and may help the training select flatter minima. Our experiments are conducted over several representative datasets, i.e., MNIST (LeCun et al., 1998), CIFAR-100 (Krizhevsky et al., 2009) and Multi30k (Elliott et al., 2016), and network structures, i.e., fully-connected neural networks, ResNet-20 (He et al., 2016) and transformer (Vaswani et al., 2017), thus our conclusion is a rather general result.

Finally, we theoretically show that, at a point close to a minimum, the covariance matrix of the noise induced by dropout and the Hessian matrix of the loss landscape is similar in the sense of the expectation with respect to the dropout randomness. The similarity between covariance and Hessian is consistent with experiments, i.e., the inverse variance-flatness relation and Hessian-variance alignment relation.

2 Related works

Dropout is proposed as a simple way to prevent neural networks from overfitting, and thus improving the generalization of the network (Hinton et al., 2012; Srivastava et al., 2014). Many works aim to find an explicit form of dropout. McAllester (2013) presents PAC-Bayesian bounds, and Wan et al. (2013), Mou et al. (2018) derive Rademacher generalization bounds. These results show that the reduction of complexity brought by dropout is $O(p)$, where p is the probability of keeping an element in dropout. Mianjy and Arora (2020) show that dropout training with logistic loss achieves ϵ-suboptimality in test error in $O(1/\epsilon)$ iterations. All of the above works need specific settings, such as norm assumptions and logistic loss, and they only give a rough estimate of the generalization error bound, which usually consider the worst case. However, it is not clear what is the characteristic of the dropout training process and how to bridge the training with the generalization. In this work, we show that dropout noise has a special structure, which closely relates with the loss landscape. The structure of the effective noise induced by the dropout may be a key reason why dropout can find solutions with better generalization.

Some works attribute the improvement in flatness to the similarity between the covariance matrix and the Hessian matrix of the loss function of SGD (Papyan, 2018, 2019). For example, Feng and Tu (2021) investigate the inverse variance-flatness relation for SGD and Zhu et al. (2018) study the Hessian-variance alignment for SGD.

3 Preliminary

3.1 Deep Neural Networks

Consider L-layer ($L \geq 2$) fully-connected DNNs with a general differentiable activation function. We regard the input as the 0th layer and the output as the Lth layer. Let m_l be the number of neurons in the lth layer. In particular, $m_0 = d$ and $m_L = d'$. For any $i, k \in \mathbb{N}$ and $i < k$, we denote $[i : k] = \{i, i+1, \ldots, k\}$. In particular, we denote $[k] := \{1, 2, \ldots, k\}$. Given weights $W^{[l]} \in \mathbb{R}^{m_l \times m_{l-1}}$ and bias $b^{[l]} \in \mathbb{R}^{m_l}$ for $l \in [L]$, we define the collection of parameters $\boldsymbol{\theta}$ as a $2L$-tuple (an ordered list of $2L$ elements) whose elements are matrices or vectors

$$\boldsymbol{\theta} = \left(\boldsymbol{\theta}|_1, \cdots, \boldsymbol{\theta}|_L\right) = \left(W^{[1]}, b^{[1]}, \ldots, W^{[L]}, b^{[L]}\right),$$

where the lth layer parameters of $\boldsymbol{\theta}$ is the ordered pair $\boldsymbol{\theta}|_l = \left(W^{[l]}, b^{[l]}\right), \quad l \in [L]$. We may misuse of notation and identify $\boldsymbol{\theta}$ with its vectorization $\mathrm{vec}(\boldsymbol{\theta}) \in \mathbb{R}^M$ with $M = \sum_{l=0}^{L-1}(m_l + 1)m_{l+1}$.

Given $\boldsymbol{\theta} \in \mathbb{R}^M$, the neural network function $\boldsymbol{f_\theta}(\cdot)$ is defined recursively. First, we write $\boldsymbol{f}_{\boldsymbol{\theta}}^{[0]}(\boldsymbol{x}) = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^d$. Then for $l \in [L-1]$, $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}$ is defined recursively as $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x}) = \sigma(W^{[l]}\boldsymbol{f}_{\boldsymbol{\theta}}^{[l-1]}(\boldsymbol{x}) + b^{[l]})$. Finally, we denote

$$\boldsymbol{f_\theta}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{f}_{\boldsymbol{\theta}}^{[L]}(\boldsymbol{x}) = W^{[L]}\boldsymbol{f}_{\boldsymbol{\theta}}^{[L-1]}(\boldsymbol{x}) + b^{[L]}.$$

For notational simplicity, we denote

$$\boldsymbol{f}_{\boldsymbol{\theta}}^{j}(\boldsymbol{x}_i) = \boldsymbol{W}_j^{[L]} \boldsymbol{f}_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i),$$

where $\boldsymbol{W}_j^{[L]} \in \mathbb{R}^{m_L}$ is the jth column of $\boldsymbol{W}^{[L]}$, and $\boldsymbol{f}_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i)$ is the jth element of vector $\boldsymbol{f}_{\boldsymbol{\theta}}^{[L-1]}(\boldsymbol{x}_i)$.

3.2 Loss function

The training data set is denoted as $S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y}_i \in \mathbb{R}^{d'}$. For simplicity, here we assume an unknown function $\boldsymbol{y}$ satisfying $\boldsymbol{y}(\boldsymbol{x}_i) = \boldsymbol{y}_i$ for $i \in [n]$. The empirical risk reads as

$$R_S(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta}), \boldsymbol{y}(\boldsymbol{x}_i)) = \mathbb{E}_S \ell(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{y}),$$

where the expectation $\mathbb{E}_S h(\boldsymbol{x}) := \frac{1}{n} \sum_{i=1}^{n} h(\boldsymbol{x}_i)$ for any function $h : \mathbb{R}^d \to \mathbb{R}$ and the loss function $\ell(\cdot, \cdot)$ is differentiable and the derivative of ℓ with respect to its first argument is denoted by $\nabla \ell(\boldsymbol{y}, \boldsymbol{y}^*)$. The error with respect to data sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ reads as

$$\epsilon(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \boldsymbol{y}_i.$$

For notational simplicity, we denote $\epsilon(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) = \epsilon_{\boldsymbol{\theta},i}$.

3.3 Dropout

For $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x}) \in \mathbb{R}^{m_l}$, we sample a scaling vector $\boldsymbol{\eta} \in \mathbb{R}^{m_l}$ with independent random coordinates,

$$(\boldsymbol{\eta})_k = \begin{cases} \frac{1-p}{p} & \text{with probability } p \\ -1 & \text{with probability } 1-p, \end{cases}$$

where $k \in [m_l]$ indexes a coordinate of $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x})$. Note that η is a zero mean random variable. We then apply dropout by computing

$$\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{[l]}(\boldsymbol{x}) = (\mathbf{1} + \boldsymbol{\eta}) \odot \boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x}),$$

and using $\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{[l]}(\boldsymbol{x})$ instead of $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x})$. Here we use $\odot$ for the Hadamard product of two matrices of the same dimension. With slight abuse of notation, we let $\boldsymbol{\eta}$ denote the collection of such vectors over all layers. $\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x})$ denotes the output of model $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ on input $\boldsymbol{x}$ using dropout noise $\boldsymbol{\eta}$. $R_S^{\mathrm{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ denotes the empirical risk with respect to network with dropout layer $\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}$, i.e.,

$$R_S^{\mathrm{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x}_i), \boldsymbol{y}(\boldsymbol{x}_i)) = \mathbb{E}_S \ell(\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x}), \boldsymbol{y}).$$

3.4 Randomness induced by dropout

3.4.1 Random trajectory data

The training process of neural networks are usually divided into two phases, fast convergence and exploration phase (Shwartz-Ziv and Tishby, 2017). In this work, we follow the experimental scheme in Feng and Tu (2021) to show the similarity between dropout and SGD. This can be understood by frequency principle (Xu et al., 2019, 2020; Zhang et al., 2021), which states that DNNs fast learn low-frequency components but slowly learn high-frequency ones.

We collect parameter sets $S_{para} = \{\boldsymbol{\theta}_i\}_{i=1}^{N}$ from N consecutive training steps in the exploration phase, where $\boldsymbol{\theta}_i$ is the network parameter set at ith sample point.

3.4.2 Random gradient data

We often need larger time interval for enough sampling to estimate the covariance accurately. Although the network loss is small, compared with the initial sampling parameters, the network parameters could have large changes during the long-time sampling. Therefore, much extra noise

may be induced. Meanwhile, for dropout, it is difficult to get a small loss value on large networks and datasets, therefore, model parameters often have large fluctuations during the sampling. To overcome this problem, we propose a more appropriate sampling method to avoid additional noise caused by sampling parameters in a large time interval. We train the network until the loss is small enough and then freeze the training. We sample N gradients of the loss function w.r.t. the parameters with different dropout variables, i.e., $S_{grad} = \{g_i\}_{i=1}^N$. In each sample, the dropout rate is fixed. In this way, we can get the noise structure of dropout without being affected by parameter changes caused by long-term training.

3.5 Inverse variance-flatness relation

We study the inverse variance flatness relation for both random trajectory data and random gradient data. For convenience, we denote data as S and its covariance as Σ.

3.5.1 Variance vs. interval flatness

The definitions of variance and interval flatness are as follows:

Definition 1 (**Variance of data at an eigen direction**). *For data S and its covariance Σ, by denoting $\lambda_i(\Sigma)$ as the ith eigenvalue of Σ, we write $\lambda_i(\Sigma)$ as the variance of the data at the corresponding eigen direction.*

Definition 2 (**Interval flatness**). [2] *For a specific solution θ_0^*, the loss function profile L_v along the direction v is:*

$$L_v(\delta\theta) \equiv L(\theta_0^* + \delta\theta v),$$

where $\delta\theta$ represents the distance moved in the v direction. The interval flatness F_v is defined as the width of the region within which $L_v(\delta\theta) \leq 2L_v(0)$. We determine F_v by finding two closest points $\theta_v^l < 0$ and $\theta_v^r > 0$ on each side of the minimum that satisfy $L_v(\theta_v^l) = L_v(\theta_v^r) = 2L_v(0)$. The interval flatness is defined as:

$$F_v \equiv \theta_v^r - \theta_v^l. \tag{1}$$

Remark. *The experiments show that the result is not sensitive to the selection of the pre-factor 2. A larger value of F_v means a flatter landscape in the direction v.*

Denote $\lambda_i(\Sigma)$ as the ith eigenvalue of Σ, and denote its corresponding eigen-vector as $v_i(\Sigma)$. The interval flatness of the loss landscape in the direction $v_i(\Sigma)$ is denoted as $F_{v_i(\Sigma)}$. We then experimentally explore the relation of $\{\lambda_i(\Sigma), F_{v_i(\Sigma)}\}_{i=1}^N$.

3.5.2 Projected variance vs. Hessian flatness

The definitions of projected variance and Hessian flatness are as follows:

Definition 3 (**Projected variance**). *For a given direction $v \in \mathbb{R}^{d_\theta}$ and a parameter set $S = \{\theta_i\}_{i=1}^{n_S}$, where $\theta_i \in \mathbb{R}^{d_\theta}$, the inner product of v and θ_i is denoted by $\text{Proj}_v(\theta_i) := \theta_i^T v$, then we can define the projected variance at direction v with respect to the sample set S as follows,*

$$\text{Var}(\text{Proj}_v(S)) = \frac{\sum_{i=1}^{n_S}(\text{Proj}_v(\theta_i) - \mu)^2}{n_S},$$

where μ is the mean value of $\{\text{Proj}_v(\theta_i)\}_{i=1}^{n_S}$.

Definition 4 (**Hessian flatness**). *For Hessian matrix H, by denoting $\lambda_i(H)$ as the ith eigenvalue of H, we write $\lambda_i(H)$ as the Hessian flatness.*

To obtain the variance induced by the dropout at a fixed position θ, we propose another way to characterize the inverse variance-flatness relation. For given data S and Hessian matrix H, we experimentally explore the relation of $\{\text{Var}(\text{Proj}_{v_i(H)}(S)), \lambda_i(H)\}_{i=1}^N$, where $\lambda_i(H)$ and $v_i(H)$ is the ith eigenvalue and eigenvector of H, respectively.

---

[2]This definition is also used in Feng and Tu (2021)

Table 1: Three types of experiments explain why dropout finds flat minima.

	DROPOUT COVARIANCE Σ	
	TRAJECTORY VARIANCE Σ_t	GRADIENT VARIANCE Σ_g
INTERVAL FLATNESS $F_{\boldsymbol{v}}$	$\{\lambda_i(\Sigma), F_{\boldsymbol{v}_i(\Sigma)}\}$, FIG. 2	
HESSIAN FLATNESS $\lambda(H)$	$\{\mathrm{Var}(\mathrm{Proj}_{\boldsymbol{v}_i(H)}(S)), \lambda_i(H)\}$, FIG. 3	
	$\backslash$	ALIGNMENT: $Tr(H\Sigma_g)$, FIG. 4.

3.6 Hessian-variance alignment

Similar to Zhu et al. (2018), we quantify the alignment between the noise structure and the curvature of loss landscape by

$$T_i = \mathrm{Tr}(H_i \Sigma_i),$$

where Σ_i is the ith-step covariance matrix of dropout layers and H_i is the Hessian matrix of the loss landscape at network parameters of the ith-step.

4 Experimental setup

To understand the effect of dropout, we train a number of networks with different structures. We consider the following types of neural networks: 1) Fully-connected neural networks (FNNs) trained by MNIST (LeCun et al., 1998). 2) Convolutional neural networks (CNNs) trained by CIFAR-10 (Krizhevsky et al., 2009). 3) Deep residual neural networks (ResNets) (He et al., 2016) trained by CIFAR-100 (Krizhevsky et al., 2009). 4) Transformer (Vaswani et al., 2017) trained by Multi30k (Elliott et al., 2016). The loss of all our experiments is cross entropy loss.

It is worth noting that, to avoid the influence of SGD in our experiments, all our networks are trained using GD, so it is difficult for us to verify on larger datasets such as ImageNet.

The detailed experimental setup can be found in Appendix A.

5 Dropout finds flatter minima

Dropout is almost ubiquitous in training deep networks. It is interesting and important to understand what makes dropout improve the generalization of training neural networks. Inspired by the study of SGD (Keskar et al., 2016), we explore the flatness of the minima found by dropout.

We adopt the method of Li et al. (2017) in this work as follows. To obtain a direction for a network with parameters $\boldsymbol{\theta}$, we begin by producing a random Gaussian direction vector $\boldsymbol{d}$ with dimensions compatible with $\boldsymbol{\theta}$. Then, we normalize each filter in $\boldsymbol{d}$ to have the same norm of the corresponding filter in $\boldsymbol{\theta}$. For FNN, each layer can be regarded as a filter. The normalization process is equivalent to normalizing the layer. For CNN, each convolution kernel may have multiple filters. Each filter is normalized individually. In other words, we make the replacement $\boldsymbol{d}_{i,j} \leftarrow \frac{\boldsymbol{d}_{i,j}}{\|\boldsymbol{d}_{i,j}\|} \|\boldsymbol{\theta}_{i,j}\|$, where $\boldsymbol{d}_{i,j}, \boldsymbol{\theta}_{i,j}$ represent the jth filter of the ith layer of the random direction $\boldsymbol{d}$ and the network parameters $\boldsymbol{\theta}$, and $\|\cdot\|$ denotes the Frobenius norm. It should be noted that j is not the index of the weight, but the filter. We use $f(\alpha) = L(\boldsymbol{\theta} + \alpha\boldsymbol{d})$ to characterize the loss landscape around the minima obtained with dropout layers $\boldsymbol{\theta}_{\boldsymbol{d}}^*$ and without dropout layer $\boldsymbol{\theta}^*$.

For all network structures shown in Fig. 1, dropout can improve the generalization of the network and find a flatter minimum. In Fig. 1(a, b), for both networks trained with and without dropout layers, the training loss values are all closed to zero, but their flatness and generalization are still different. In Fig. 1(c, d), due to the complexity of the dataset, i.e., CIFAR-100 and Multi30k, and networks, i.e., ResNet-20 and transformer, networks with dropout layers does not achieve very small training error but the ones with dropout find flatter minima with much better generalization.

(a) flatness of FNN   (b) flatness of vgg-9   (c) flatness of ResNet-20   (d) flatness of transformer

Figure 1: The 1D visualization of solutions of different network structures obtained with or without dropout layers. (a) The FNN is trained on MNIST dataset. The test accuracy for model with dropout layers is 98.7% while 98.1% for model without dropout layers. (b) The vgg-9 network is trained on CIFAR-10 dataset using the first 2048 examples as training dataset. The test accuracy for model with dropout layers is 60.6% while 59.2% for model without dropout layers. (c) The ResNet-20 network is trained on CIFAR-100 dataset using all examples as training dataset. The test accuracy for model with dropout layers is 54.7% while 34.1% for model without dropout layers. (d) The transformer is trained on the Multi30k dataset using the first 2048 examples as training dataset. The test accuracy for model with dropout layers is 49.33% while 34.73% for model without dropout layers.

Next, we utilize three methods to examine the relation between the covariance of the noise induced by the dropout randomness and the Hessian of the loss landscape, as summarized in Table 1.

6   Inverse variance-flatness relation

Similar to SGD, the effect of dropout can be equivalent to imposing a specific noise on the gradient. A random noise, such as isotropic noise, can help the training escape local minima, but can not robustly improve generalization (An, 1996; Zhu et al., 2018). The noise induced by the dropout should have certain properties that can lead the training to good minima.

In this section, we show that the noise induced by the dropout satisfies the inverse variance-flatness relation, that is, the noise variance is larger along the sharper direction of the loss landscape at a minimum. The landscape-dependent structure helps the training escape sharp minima.

6.1   Variance vs. interval flatness

We use the principal component analysis (PCA) to study the weight variations when the accuracy is nearly 100%. For FNNs, networks are trained on MNIST with the first 10000 examples as the training set for computational efficiency. For ResNets, networks are trained on CIFAR-100 with 50000 examples as the training set. For the transformer structure, the network is trained by Multi30k (Vaswani et al., 2017). The networks are trained with full batch for different learning rates and dropout rates under the same random seed (that is, with the same initialization parameters). When the loss is small enough, we sample the parameters or gradients of parameters N times ($N = 3000$ in this experiment) and use the method introduced in Section 3.4 to construct covariance matrix Σ by the weights S_{para} or gradients S_{grad} of specific network parameters mentioned in Section 4. The PCA is done for the covariance matrix Σ. We then compute the interval flatness of the loss function landscape at eigen-directions, i.e., $\{F_{\boldsymbol{v}_i(\Sigma)}\}_{i=1}^N$. Note that the PCA spectrum $\{\lambda_i(\Sigma)\}_{i=1}^N$ indicate the variance of weights S_{para} or gradients S_{grad} at corresponding eigen-directions.

As shown in Fig. 2, for different learning rates and dropout rates, there is an inverse relationship between the interval flatness of the loss function landscape $\{F_{\boldsymbol{v}_i(\Sigma)}\}_{i=1}^N$ and the dropout variance, i.e., the PCA spectrum $\{\lambda_i(\Sigma)\}_{i=1}^N$. We can approximately see a power-law relationship between $\{F_{\boldsymbol{v}_i(\Sigma)}\}_{i=1}^N$ and $\{\lambda_i(\Sigma)\}_{i=1}^N$. More detailed, for the small flatness part, the variance of noise induced by dropout is generally large, which indicates that the noise induced by dropout has larger variance in sharp directions; for the large flatness part, as the loss landscape gets flatter, the linear relationship is more obvious, and we can see a clearer asymptotic behavior in the results. Overall, we can observe the negative correlation between the variance and flatness in Fig. 2.

(a) FNN, datasets S sampled from parameters

(b) ResNet-20, datasets S sampled from parameters

(c) Transformer, datasets S sampled from parameters

(d) FNN, datasets S sampled from gradients of parameters

(e) ResNet-20, datasets S sampled from gradients of parameters

(f) Transformer, datasets S sampled from gradients of parameters

Figure 2: The inverse relation between the variance $\{\lambda_i(\Sigma)\}_{i=1}^N$ and the interval flatness $\{F_{\boldsymbol{v}_i(\Sigma)}\}_{i=1}^N$ for different choices of dropout rate p and learning rate lr with different network structures. The PCA is done for different datasets S sampled from parameters for the top line and sampled from gradients of parameters for the bottom line. The dash lines give the approximate slope of the scatter.

6.2 Projected variance vs. Hessian flatness

The eigenvalues of the Hessian of the loss at a minimum are also often used to indicate the flatness. A large eigenvalue corresponds to a sharper direction. In this section, we study the relationship between eigenvalues of Hessian H of loss landscape at the end point of training and the variances of dropout at corresponding eigen-directions. As mentioned in the Preliminary, we sample the parameters or gradients of parameters 1000 times, that is $N = 1000$. For each eigen-direction $\boldsymbol{v}_i$ of Hessian H, we project the sampled parameters or the gradients of sampled parameter to direction $\boldsymbol{v}_i$ by inner product, denoted by $\text{Proj}_{\boldsymbol{v}_i}(S)$. Then, we compute the variance of the projected data, i.e., $\text{Var}(\text{Proj}_{\boldsymbol{v}_i}(S))$.

As shown in Fig. 3, we find that there is also a power-law relationship between $\{\lambda_i(H)\}_{i=1}^D$ and $\{\text{Var}(\text{Proj}_{\boldsymbol{v}_i}(S))\}_{i=1}^D$ for different dropout rates and learning rates, no matter S is sampled from parameters or gradients of parameters. The positive correlation between the eigenvalue and the projection variance show the structure of the dropout noise, which helps the network escape the bad minima. At the same time, as shown in Figs. 2 and 3, we can see that gradient sampling has a more clear linear structure than that of parameter sampling.

7 Hessian-variance alignment

In this section, we study the alignment between the Hessian and the random gradient covariance at each training step, i.e., Hessian-variance alignment. Note that the training is performed by GD without dropout. At step i, we sample the gradients of parameters $\{\boldsymbol{g}_i^j\}_{j=1}^N$ by tentatively adding a dropout layer between the hidden layers. For each step i, we the compute $\text{Tr}(H_i\Sigma_i)$, where H_i is the Hessian of the loss at the parameter set at step i and Σ_i is the covariance of $\{\boldsymbol{g}_i^j\}_{j=1}^N$.

In order to show the anisotropic structure, we construct the isotropic noise for comparison, i.e., $\bar{\Sigma}_i = \frac{\text{Tr}\,\Sigma_i}{D}I$ of the covariance matrix Σ_i, where D is the number of parameters. In our experiments, $D = 2500$. As shown in Fig. 4, in the whole training process under different learning rates and dropout rates, $\text{Tr}(H_i\Sigma_i)$ is much larger than $\text{Tr}(H_i\bar{\Sigma}_i)$, indicating the anisotropic structure of dropout noise and its high alignment with the Hessian matrix.

7

(a) FNN, FNN, datasets S sampled from parameters

(b) ResNet-20, datasets S sampled from parameters

(c) Transformer, datasets S sampled from parameters

(d) FNN, datasets S sampled from gradients of parameters

(e) ResNet-20, datasets S sampled from gradients of parameters

(f) Transformer, datasets S sampled from gradients of parameters

Figure 3: The relation between the variance $\{\mathrm{Var}(\mathrm{Proj}_{\boldsymbol{v}_i}(S))\}_{i=1}^{D}$ and the eigenvalue $\{\lambda_i(H)\}_{i=1}^{D}$ for different choices of dropout rate p and learning rate lr with different network structures. The projection is done for different datasets S sampled from parameters for the top line and sampled from gradients of parameters for the bottom line. The dash lines give the approximate slope of the scatter.

Figure 4: Comparison between $\mathrm{Tr}(H_i \Sigma_i)$ and $\mathrm{Tr}(H_i \bar{\Sigma}_i)$ in each training epoch i for different choices of dropout rate p and learning rate lr. The FNN is trained on MNIST dataset using the first 10000 examples as training dataset. The solid and the dotted lines represent the value of $\mathrm{Tr}(H_i \Sigma_i)$ and $\mathrm{Tr}(H_i \bar{\Sigma}_i)$, respectively.

8   Theoretical analysis

In this section, we summarize key theoretical results on the similarity between Hessian and covariance matrices under dropout regularization. The proofs are in the Appendix B. We first summarize the specific settings and the assumptions required for our theoretical results:

Setting 1 (Dropout structure). *Consider a L-layer ($L \geq 2$) fully-connected DNN has only one dropout layer after the $L-1$th layer of the network,*

$$\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{drop}(\boldsymbol{x}) = \boldsymbol{W}^{[L]}(\mathbf{1}+\boldsymbol{\eta}) \odot \boldsymbol{f}_{\boldsymbol{\theta}}^{[L-1]}(\boldsymbol{x}) + \boldsymbol{b}^{[L]}.$$

8

Setting 2 (Loss function). *Take the mean squared error (MSE) as our loss function,*

$$R_S(\boldsymbol{\theta}) = \mathbb{E}_S \ell(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{y}) = \frac{1}{2n} \sum_{i=1}^{n} (\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta}) - \boldsymbol{y}_i)^2.$$

Setting 3 (Network structure). *Take the piece-wise linear function as our activation function. For convenience, we further set that the model output is an one-dimensional vector, i.e. $m_L = 1$.*

Assumption 1. *We examine the loss landscape after training reaches a stable stage, so we assume that the gradient of the loss function in the average sense is small enough, i.e.*

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\eta}} R_S^{\mathrm{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) \approx \mathbf{0}.$$

Under the above assumptions and settings, we can theoretically calculate the Hessian matrix and covariance matrix of the loss function as follows.

Theorem 1 (Hessian matrix with dropout regularization). *Based on the Setting 1-3 and Assumption 1, the Hessian matrix of the loss function with respect to $\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x})$ can be written in the mean sense as:*

$$H(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\eta}} \nabla_{\boldsymbol{\theta}}^2 R_S^{\mathrm{drop}} = \frac{1}{n} \sum_{i=1}^{n} \left[\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \frac{1-p}{p} \sum_{j=1}^{m_{L-1}} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \right].$$

Theorem 2 (Covariance matrix with dropout regularization). *Based on the Setting 1-3 and Assumption 1, the covariance matrix of the loss function under the randomness of dropout variable $\boldsymbol{\eta}$ and data $\boldsymbol{x}$ can be written as:*

$$\Sigma_{\boldsymbol{\theta}}^{\mathrm{drop}} = \frac{1}{n} \sum_{i=1}^{n} \left[a_i \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + b_i \frac{1-p}{p} \sum_{j=1}^{m_{L-1}} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \right],$$

where $a_i = 2\mathbb{E}_{\boldsymbol{\eta}} \ell(\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$, $b_i = 2\ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$.

Proposition 1. *Based on the Setting 1-3 and Assumption 1, we further restrict the problem to a binary classification problem, i.e. $\boldsymbol{y}_i \in \{0, 1\}$, $\forall i \in [n]$, and assume the model output $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \in [\delta, 1-\delta]$ (we can limit the network output using a threshold activation function), where δ is a small positive constant, then we have:*

(i) $\Sigma_{\boldsymbol{\theta}}^{\mathrm{drop}} \succeq \delta^2 H(\boldsymbol{\theta})$, almost everywhere in $\mathbb{R}^M$, M is the dimension of $\boldsymbol{\theta}$;

(ii) For any $\epsilon > 0$, and a network parameter $\boldsymbol{\theta} \in \Omega = \{\boldsymbol{\theta} : \mathbb{E}_{\boldsymbol{\eta}} \ell(\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x}_i), \boldsymbol{y}_i) \leq \frac{(\delta+\epsilon)^2}{2}, \ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) \leq \frac{(\delta+\epsilon)^2}{2}, \forall i \in [n]\}$, we have $\Sigma_{\boldsymbol{\theta}}^{\mathrm{drop}} \preceq (\delta + \epsilon)^2 H(\boldsymbol{\theta})$ almost everywhere in Ω.

Remark. *$A \preceq B$ means that $(B - A)$ is semi-positive definite.*

Remark. *The results in Proposition 1 are consist with the results under SGD setting studied in Papyan (2018, 2019); Zhu et al. (2018).*

From above analysis, we can see the Hessian and the covariance are very similar. Especially, when the training is approaching the end, the error of all samples has similar magnitude, then, H and $\Sigma_{\boldsymbol{\theta}}^{\mathrm{drop}}$ has an approximately linear relation.

9 Conclusion and discussion

In this work, we find that dropout training selects flatter minima compared with standard gradient descent training. We further show inverse variance-flatness relation and Hessian-variance alignment. These two relations may help the training select flatter minima and leads the training to good generalization. We then theoretically show the similarity between the Hessian and covariance to further support the goodness of dropout. The dropout and the SGD are common in sharing the these two relations. As a starting point, our work shows a promising and reasonable direction for understanding the stochastic training of neural networks.

References

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

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (2014) 1929–1958.

M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, 2019, pp. 6105–6114.

D. P. Helmbold, P. M. Long, On the inductive bias of dropout, The Journal of Machine Learning Research 16 (2015) 3403–3454.

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

Y. Feng, Y. Tu, The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima, Proceedings of the National Academy of Sciences 118 (2021).

Z. Zhu, J. Wu, B. Yu, L. Wu, J. Ma, The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects, arXiv preprint arXiv:1803.00195 (2018).

C. Wei, S. Kakade, T. Ma, The implicit and explicit regularization effects of dropout, in: International Conference on Machine Learning, PMLR, 2020, pp. 10181–10192.

B. Neyshabur, S. Bhojanapalli, D. McAllester, N. Srebro, Exploring generalization in deep learning, arXiv preprint arXiv:1706.08947 (2017).

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

A. Krizhevsky, et al., Learning multiple layers of features from tiny images (2009).

D. Elliott, S. Frank, K. Sima'an, L. Specia, Multi30k: Multilingual english-german image descriptions, in: 5th Workshop on Vision and Language, Association for Computational Linguistics (ACL), 2016, pp. 70–74.

K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

D. McAllester, A pac-bayesian tutorial with a dropout bound, arXiv preprint arXiv:1307.2118 (2013).

L. Wan, M. Zeiler, S. Zhang, Y. Lecun, R. Fergus, Regularization of neural networks using dropconnect, in: In Proceedings of the International Conference on Machine learning, Citeseer, 2013.

W. Mou, Y. Zhou, J. Gao, L. Wang, Dropout training, data-dependent regularization, and generalization bounds, in: International conference on machine learning, PMLR, 2018, pp. 3645–3653.

P. Mianjy, R. Arora, On convergence and generalization of dropout training, Advances in Neural Information Processing Systems 33 (2020).

V. Papyan, The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size, arXiv preprint arXiv:1811.07062 (2018).

V. Papyan, Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians, arXiv preprint arXiv:1901.08244 (2019).

R. Shwartz-Ziv, N. Tishby, Opening the black box of deep neural networks via information, arXiv preprint arXiv:1703.00810 (2017).

Z.-Q. J. Xu, Y. Zhang, Y. Xiao, Training behavior of deep neural network in frequency domain, in: International Conference on Neural Information Processing, Springer, 2019, pp. 264–274.

Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, Z. Ma, Frequency principle: Fourier analysis sheds light on deep neural networks, Communications in Computational Physics 28 (2020) 1746–1767.

Y. Zhang, T. Luo, Z. Ma, Z.-Q. J. Xu, A linear frequency principle model to understand the absence of overfitting in neural networks, Chinese Physics Letters 38 (2021) 038701.

H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein, Visualizing the loss landscape of neural nets, arXiv preprint arXiv:1712.09913 (2017).

G. An, The effects of adding noise during backpropagation training on a generalization performance, Neural computation 8 (1996) 643–674.

D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR (Poster), 2015.

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

Checklist

1. For all authors...

    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

    (b) Did you describe the limitations of your work? [Yes] See Assumptions and settings in Section 8.

    (c) Did you discuss any potential negative societal impacts of your work? [N/A]

    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

    (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 8.

    (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B

3. If you ran experiments...

    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the material.

    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A

    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The provider information Will be shown in Acknowledgement.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

    (a) If your work uses existing assets, did you cite the creators? [Yes]

    (b) Did you mention the license of the assets? [No] The datatsets we used are well known.

    (c) Did you include any new assets either in the supplemental material or as a URL? [No]

    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

    (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

      (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

      (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A   Detailed experimental setup

For Fig. 1(a), we use the FNN with size $784 - 1024 - 1024 - 10$. We add dropout layers behind the first and the second layers with dropout rate of 0.8 and 0.5, respectively. We train the network using default Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0001.

For Fig. 1(b), we use vgg-9 (Simonyan and Zisserman, 2014) to compare the loss landscape flatness w/o dropout layers. For experiment with dropout layers, we add dropout layers after the pooling layers, the dropout rates of dropout layers are 0.8. Models are trained using GD with Nesterov momentum, training-size 2048 for 300 epochs. The learning rate is initialized at 0.1, and divided by a factor of 10 at epochs 150, 225 and 275. We only use the first 2048 examples for training to compromise with the computational burden.

For Fig. 2(a, d), Fig. 3(a, d), Fig. 4, we use the FNN with size $784 - 50 - 50 - 10$. We train the network using GD with the first 10,000 training data as the training set. We add a dropout layer behind the second layer. The dropout rate and learning rate are specified and unchanged in each experiment. We only consider the parameter matrix corresponding to the weight and the bias of the fully-connected layer between two hidden layers.

For Fig. 1(c), Fig. 2(b, e), Fig. 3(b, e), we use ResNet-20 (He et al., 2016) to compare the loss landscape flatness w/o dropout layers. For experiment with dropout layers, we add dropout layers after the convolutional layers, the dropout rates of dropout layers are 0.8. We only consider the parameter matrix corresponding to the weight of the first convolutional layer of the first block of the ResNet-20. Models are trained using GD, training-size 50000 for 1200 epochs. The learning rate is initialized at 0.01. Since the Hessian calculation of ResNet takes much time, for the ResNet experiment, we only perform it at a specific dropout rate and learning rate.

For Fig. 1(d), Fig. 2(c, f), Fig. 3(c, f), we use transformer (Vaswani et al., 2017) with $d_{model} = 50, d_k = d_v = 20, d_{ff} = 256, h = 4, N = 3$, the meaning of the parameters is consistent with the original paper. In order to calculate the Hessian matrix and eigendecomposition more accurately and quickly, we reasonably reduce the number of network parameters. We only consider the parameter matrix corresponding to the weight of the fully-connected layer whose output is queries in the Multi-Head Attention layer of the first block of the decoder. For experiment with dropout layers, we apply dropout to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. The dropout rates of dropout layers are 0.9. For the English-German translation problem, we use the cross-entropy loss with label smoothing trained by full-batch Adam based on the Multi30k dataset. The learning rate strategy is the same as that in the article. The warm up step is 4000 epochs, the training step is 10000 epochs. We only use the first 2048 examples for training to compromise with the computational burden.

B   Derivations and Proofs for Main Paper

B.1   Proof of Theorem 1

Theorem (Theorem 1: Hessian matrix with dropout regularization). *Based on the Setting 1-3 and Assumption 1, the Hessian matrix of the loss function with respect to $\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x})$ can be written in the mean sense as:*

$$H(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\eta}} \nabla_{\boldsymbol{\theta}}^2 R_S^{\mathrm{drop}} = \frac{1}{n} \sum_{i=1}^{n} \left[\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \frac{1-p}{p} \sum_{j=1}^{m_{L-1}} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \right].$$

Proof. We first compute the Hessian matrix after taking expectation with respect to the dropout variable,

$$\mathbb{E}_{\boldsymbol{\eta}}\nabla_{\boldsymbol{\theta}}^2 R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \nabla_{\boldsymbol{\theta}}^2 R_S(\boldsymbol{\theta}) + \frac{1-p}{2np} \sum_{i=1}^{n} \sum_{j=1}^{m_{L-1}} \nabla_{\boldsymbol{\theta}}^2 (\boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i))^2. \tag{2}$$

The first and second terms on the RHS of the Equ.(2) are as follows,

$$\nabla_{\boldsymbol{\theta}}^2 R_S(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \boldsymbol{y}_i) \cdot \nabla_{\boldsymbol{\theta}}^2 \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \right)$$

$$\frac{1-p}{2np} \sum_{i=1}^{n} \sum_{j=1}^{m_{L-1}} \nabla_{\boldsymbol{\theta}}^2 (\boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i))^2 = \frac{1-p}{np} \sum_{i=1}^{n} \sum_{j=1}^{m_{L-1}} \left(\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) + \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \cdot \nabla_{\boldsymbol{\theta}}^2 \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \right).$$

Note that for linear activate function, $\nabla_{\boldsymbol{\theta}}^2 \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \nabla_{\boldsymbol{\theta}}^2 \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) = \boldsymbol{0}, \ a.e. \ \forall i \in [n], \forall j \in [m]$, we have

$$\nabla_{\boldsymbol{\theta}}^2 R_S(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

$$\frac{1-p}{2np} \sum_{i=1}^{n} \sum_{j=1}^{m_{L-1}} \nabla_{\boldsymbol{\theta}}^2 (\boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i))^2 = \frac{1-p}{np} \sum_{i=1}^{n} \sum_{j=1}^{m_{L-1}} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i).$$

Thus the Equ.(2) can be rewritten as

$$\mathbb{E}_{\boldsymbol{\eta}}\nabla_{\boldsymbol{\theta}}^2 R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \frac{1}{n} \sum_{i=1}^{n} \left(\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \frac{1-p}{p} \sum_{j=1}^{m} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \right).$$

$\square$

B.2 Proof of Theorem 2

Theorem (**Theorem 2: Covariance matrix with dropout regularization**). *Based on the Setting 1-3 and Assumption 1, the covariance matrix of the loss function under the randomness of dropout variable $\boldsymbol{\eta}$ and data $\boldsymbol{x}$ can be written as:*

$$\Sigma_{\boldsymbol{\theta}}^{\text{drop}} = \frac{1}{n} \sum_{i=1}^{n} \left[a_i \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + b_i \frac{1-p}{p} \sum_{j=1}^{m_{L-1}} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \boldsymbol{f}_{\boldsymbol{\theta}}^j(\boldsymbol{x}_i) \right],$$

where $a_i = 2\mathbb{E}_{\boldsymbol{\eta}} \ell(\boldsymbol{f}_{\boldsymbol{\theta}, \boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$, $b_i = 2\ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$.

Proof. For simplicity, we approximate the loss function through Taylor expansion, which is also used in Wei et al. (2020),

$$\ell(\boldsymbol{f}_{\boldsymbol{\theta}, \boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x}_i), \boldsymbol{y}_i) \approx \ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) + (\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \boldsymbol{y}_i) \sum_{j=1}^{m_{L-1}} \boldsymbol{W}_j^{[L]}(\boldsymbol{\eta})_j \boldsymbol{f}_{\boldsymbol{\theta}, j}^{[L-1]}(\boldsymbol{x}_i),$$

where $\boldsymbol{W}_j^{[L]} \in \mathbb{R}^{m_L}$ is the jth column of $\boldsymbol{W}^{[L]}$, and $\boldsymbol{f}_{\boldsymbol{\theta}, j}^{[L-1]}(\boldsymbol{x}_i)$ is the jth element of vector $\boldsymbol{f}_{\boldsymbol{\theta}}^{[L-1]}(\boldsymbol{x}_i)$. The covariance matrix obtained using SGD under dropout regularization is

$$\Sigma_{\boldsymbol{\theta}}^{\text{drop}} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{\eta}} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{f}_{\boldsymbol{\theta}, \boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x}_i), \boldsymbol{y}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \ell(\boldsymbol{f}_{\boldsymbol{\theta}, \boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x}_i), \boldsymbol{y}_i) - \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\eta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) \nabla_{\boldsymbol{\theta}}^{\intercal} \mathbb{E}_{\boldsymbol{\eta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{\eta}} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{f}_{\boldsymbol{\theta}, \boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x}_i), \boldsymbol{y}_i) \nabla_{\boldsymbol{\theta}}^{\intercal} \ell(\boldsymbol{f}_{\boldsymbol{\theta}, \boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x}_i), \boldsymbol{y}_i).$$

Combining the properties of the dropout variable $\boldsymbol{\eta}$, we have,

$$\Sigma_{\boldsymbol{\theta}}^{\mathrm{drop}} = \frac{1}{n}\sum_{i=1}^{n}\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{f_\theta}(\boldsymbol{x}_i),\boldsymbol{y}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\ell(\boldsymbol{f_\theta}(\boldsymbol{x}_i),\boldsymbol{y}_i)$$

$$+\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\boldsymbol{\eta}}\left(\sum_{j=1}^{m_{L-1}}(\boldsymbol{\eta})_j\nabla_{\boldsymbol{\theta}}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})\sum_{j=1}^{m_{L-1}}(\boldsymbol{\eta})_j\nabla_{\boldsymbol{\theta}}^{\intercal}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})\right)$$

$$=\frac{1}{n}\sum_{i=1}^{n}\left(\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{f_\theta}(\boldsymbol{x}_i),\boldsymbol{y}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\ell(\boldsymbol{f_\theta}(\boldsymbol{x}_i),\boldsymbol{y}_i)+\frac{1-p}{p}\sum_{j=1}^{m_{L-1}}\nabla_{\boldsymbol{\theta}}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})\nabla_{\boldsymbol{\theta}}^{\intercal}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})\right)$$

$$:=\frac{1}{n}\sum_{i=1}^{n}\left(\Sigma_{\boldsymbol{\theta},1}^{\mathrm{drop}}(\boldsymbol{x}_i,\boldsymbol{y}_i)+\frac{1-p}{p}\Sigma_{\boldsymbol{\theta},2}^{\mathrm{drop}}(\boldsymbol{x}_i,\boldsymbol{y}_i)\right).$$

$$(3)$$

We calculate the two terms on the RHS of the Equ.(3) separately:

$$\Sigma_{\boldsymbol{\theta},1}^{\mathrm{drop}}(\boldsymbol{x}_i,\boldsymbol{y}_i)=(\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_i))^2\cdot\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta}(\boldsymbol{x}_i),$$

$$\Sigma_{\boldsymbol{\theta},2}^{\mathrm{drop}}(\boldsymbol{x}_i,\boldsymbol{y}_i)=(\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})^2\sum_{j=1}^{m_{L-1}}\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)+\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\sum_{j=1}^{m_{L-1}}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i))^2$$

$$+2\sum_{j=1}^{m_{L-1}}\boldsymbol{\epsilon}_{\boldsymbol{\theta},i}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\cdot\nabla_{\boldsymbol{\theta}}\boldsymbol{\epsilon}_{\boldsymbol{\theta},i}\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)$$

$$=(\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})^2\sum_{j=1}^{m_{L-1}}\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)+\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\sum_{j=1}^{m_{L-1}}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i))^2$$

$$+\frac{1}{2}\sum_{j=1}^{m_{L-1}}\nabla_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})^2\nabla_{\boldsymbol{\theta}}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i))^2.$$

Under the assumption that $\nabla_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})^2=2\cdot\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{f_\theta}(\boldsymbol{x}_i),\boldsymbol{y}_i)=\mathbf{0},\forall i\in[n]$, we have

$$\Sigma_{\boldsymbol{\theta},2}^{\mathrm{drop}}(\boldsymbol{x}_i,\boldsymbol{y}_i)=(\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})^2\sum_{j=1}^{m_{L-1}}\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)+\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\sum_{j=1}^{m_{L-1}}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i))^2.$$

Thus the Equ.(3) can be rewritten as

$$\Sigma_{\boldsymbol{\theta}}^{\mathrm{drop}}=\frac{1}{n}\sum_{i=1}^{n}\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\left((\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_i))^2+\frac{1-p}{p}\sum_{j=1}^{m_{L-1}}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i))^2\right)$$

$$+\frac{1-p}{np}\sum_{i=1}^{n}\sum_{j=1}^{m_{L-1}}(\boldsymbol{\epsilon}_{\boldsymbol{\theta},i})^2\cdot\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i).$$

Note that

$$(\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_i))^2+\frac{1-p}{p}\sum_{j=1}^{m_{L-1}}(\boldsymbol{f_\theta^j}(\boldsymbol{x}_i))^2=\mathbb{E}_{\boldsymbol{\eta}}2\ell(\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x}_i),\boldsymbol{y}_i),$$

we have

$$\Sigma_{\boldsymbol{\theta}}^{\mathrm{drop}}=\frac{2}{n}\sum_{i=1}^{n}\mathbb{E}_{\boldsymbol{\eta}}\ell(\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\mathrm{drop}}(\boldsymbol{x}_i),\boldsymbol{y}_i)\cdot\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta}(\boldsymbol{x}_i)$$

$$+\frac{2(1-p)}{np}\sum_{i=1}^{n}\sum_{j=1}^{m_{L-1}}(\ell(\boldsymbol{f_\theta}(\boldsymbol{x}_i),\boldsymbol{y}_i))\cdot\nabla_{\boldsymbol{\theta}}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{f_\theta^j}(\boldsymbol{x}_i).$$

$$\square$$

B.3 Proof of Proposition 1

Proposition (**Proposition 1**). *Based on the Setting 1-3 and Assumption 1, we further restrict the problem to a binary classification problem, i.e. $y_i \in \{0, 1\}$, $\forall i \in [n]$, and assume the model output $f_\theta(x_i) \in [\delta, 1 - \delta]$ (we can limit the network output using a threshold activation function), where δ is a small positive constant, then we have:*

(i) $\Sigma_\theta^{\mathrm{drop}} \succeq \delta^2 H(\theta)$, almost everywhere in $\mathbb{R}^M$, M is the dimension of θ;

(ii) For any $\epsilon > 0$, and a network parameter $\theta \in \Omega = \{\theta : \mathbb{E}_\eta \ell(f_{\theta,\eta}^{\mathrm{drop}}(x_i), y_i) \leq \frac{(\delta+\epsilon)^2}{2}, \ell(f_\theta(x_i), y_i) \leq \frac{(\delta+\epsilon)^2}{2}, \forall i \in [n]\}$, we have $\Sigma_\theta^{\mathrm{drop}} \preceq (\delta + \epsilon)^2 H(\theta)$ almost everywhere in Ω.

Proof. The properties (i)–(ii) are direct consequences of Thm. 1, 2. $\qquad\square$