Guided Dropout

Rohit Keshari, Richa Singh, Mayank Vatsa
IIIT-Delhi, India
{rohitk, rsingh, mayank}@iiitd.ac.in

Abstract

Dropout is often used in deep neural networks to prevent over-fitting. Conventionally, dropout training invokes *random drop* of nodes from the hidden layers of a Neural Network. It is our hypothesis that a guided selection of nodes for intelligent dropout can lead to better generalization as compared to the traditional dropout. In this research, we propose "guided dropout" for training deep neural network which drop nodes by measuring the strength of each node. We also demonstrate that conventional dropout is a specific case of the proposed guided dropout. Experimental evaluation on multiple datasets including MNIST, CIFAR10, CIFAR100, SVHN, and Tiny ImageNet demonstrate the efficacy of the proposed guided dropout.

Introduction

"Better than a thousand days of diligent study is one day with a great teacher."

— Japanese proverb

Deep neural network has gained a lot of success in multiple applications. However, due to optimizing millions of parameters, generalization of Deep Neural Networks (DNN) is a challenging task. Multiple regularizations have been proposed in the literature such as $l_1 - norm$ (Nowlan and Hinton 1992), $l_2 - norm$ (Nowlan and Hinton 1992), max-norm (Srivastava et al. 2014), rectifiers (Nair and Hinton 2010), KL-divergence (Hinton, Osindero, and Teh 2006), drop-connect (Wan et al. 2013), and dropout (Hinton et al. 2012), (Srivastava et al. 2014) to regulate the learning process of deep neural networks consisting of a large number of parameters. Among all the regularizers dropout has been widely used for the generalization of DNNs.

Dropout (Hinton et al. 2012), (Srivastava et al. 2014) improves the generalization of neural networks by preventing co-adaptation of feature detectors. The working of dropout is based on the generation of a mask by utilizing $Bernoulli$ and $Normal$ distributions. At every iteration, it generates a random mask with probability $(1-\theta)$ for hidden units of the network. (Wang and Manning 2013) have proposed a Gaussian dropout which is a fast approximation of conventional dropout. (Kingma, Salimans, and Welling 2015) have proposed variational dropout to reduce the variance of Stochastic Gradients for Variational Bayesian inference (SGVB). They have shown that variational dropout is a generalization of Gaussian dropout where the dropout rates are learned.

(Klambauer et al. 2017) have proposed alpha-dropout for Scaled Exponential Linear Unit (SELU) activation function. (Ba and Frey 2013) have proposed "standout" for a deep belief neural network where, instead of initializing dropout mask using $Bernoulli$ distribution with probability p, they have adapted the dropout probability for each layer of the neural network. In addition to the conventional learning methods of dropout, (Gal and Ghahramani 2016) have utilized the Gaussian process for the deep learning models which allows estimating uncertainty of the function, robustness to over-fitting, and hyper-parameter tuning. They have measured model uncertainty by measuring the first and second moments of their approximate predictive distribution. (Gal, Hron, and Kendall 2017) have proposed "Concrete Dropout" which is a variant of dropout where concrete distribution has been utilized to generate the dropout mask. They have optimized the probability p via path-wise derivative estimator.

In the literature, methods related to dropout have been explored in two aspects: 1) sampling dropout mask from different distributions and maintaining mean of the intermediate input while dropping nodes, and 2) adapting dropout probability. However, if prior information related to nodes of a Neural Network (NN) is available, nodes can be dropped selectively in such a way that generalization of NN is improved. Therefore, in this research, we propose "strength parameter" to measure the importance of nodes and feature-map for dense NN and CNN, respectively and use it for guiding dropout regularization. Figure 1 illustrates the graphical abstract of the paper "guided dropout". For understanding the behavior of strength parameter t, a three hidden layer neural network with 8192 nodes is trained with strength parameter using Equation 2 (details discussed in the next section). After training, the accuracy is evaluated by removing low strength to high strength nodes one by one. The effect on the accuracy can be observed in Figure 2. It shows that removing up to almost 5000 low strength nodes has minimal affect on the network accuracy. Therefore, such nodes are considered to be *inactive nodes*, lying in the *inactive region*.

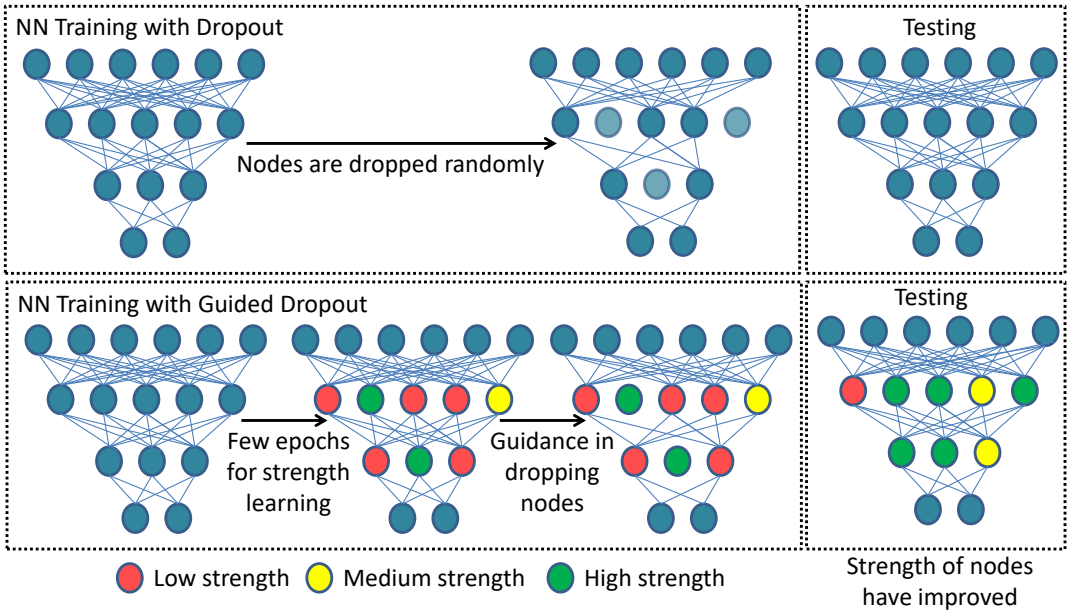

Figure 1: Illustrating of the effect of conventional dropout and proposed guided dropout. In neural network training with dropout, nodes are dropped randomly from the hidden layers. However, in the proposed guided dropout, nodes are dropped based on their strength. (Best viewed in color).

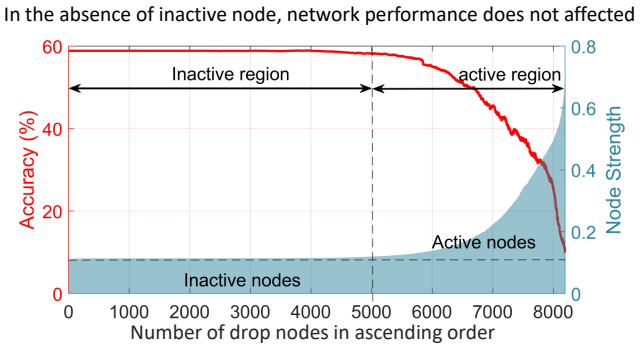

Figure 2: A Neural Network NN[8192, 3] is trained with strength parameter on CIFAR10 dataset. The bar graph represents the trained strength of the first layer of the NN. It can be observed that low strength nodes are not contributing in the performance and removing such nodes minimally affect the accuracy. Such nodes are termed as inactive nodes in the inactive region. Similarly, high strength nodes are contributing in the performance and removing such nodes affect the accuracy. Such nodes are termed as active node in the active region. (Best viewed in color).

On removing nodes with high strength, the network accuracy reduces aggressively. Such nodes are considered to be *active nodes* in the *active region*.

Our hypothesis is that in the absence of high strength nodes during training, low strength nodes can improve their strength and contribute to the performance of NN. To achieve this, while training a NN, we drop the high strength nodes in the active region and learn the network with low strength nodes. This is termed as *Guided Dropout*. As shown in Figure 1, during training the network generalizability is strengthened by "nurturing" inactive nodes. Once trained, more nodes are contributing towards making predictions thus improving the accuracy. The key contribution of this paper is: Strength parameter is proposed for deep neural networks which is associated with each node. Using this parameter, a novel guided dropout regularization approach is proposed. To the best of our knowledge, this is the first attempt to remove randomness in the mask generation process of dropout. We have also presented that conventional dropout is a special case of guided dropout, and is observed when the concept of active and inactive regions are not considered. Further, experimental and theoretical justifications are also presented to demonstrate that the proposed guided dropout performance is always equal to or better than the conventional dropout.

Proposed Guided Dropout

In dropout regularization, some of the nodes from the hidden layers are dropped at every epoch with probability $(1-\theta)$. Let l be the l^{th} layer of a network, where the value of l ranges from 0 to L, and L is the number of hidden layers in the network. When the value of l is zero, it represents the input layer, i.e. $a^0 = X$. Let the intermediate output of the network be $z^{(l)}$. Mathematically, it can be expressed as:

$$\begin{aligned} z_j^{(l+1)} &= w_{j\times i}^{(l+1)} a_i^l + b_j^{(l+1)} \\ a_j^{(l+1)} &= f(z_j^{(l+1)}) \end{aligned} \tag{1}$$

where, $i \in [1, ..., N_{in}]$, and $j \in [1, ..., N_{out}]$ are index variables for N_{in} and N_{out} at the $(l+1)^{th}$ layer, respectively. $f(.)$ is the $RELU$ activation function. The conventional dropout drops nodes randomly using $Bernoulli$ distribution and is expressed as: $\widetilde{\mathbf{a}}^{(l)} = \mathbf{r}^{(l)} \odot \mathbf{a}^{(l)}$, where $\widetilde{a}$ is the masked output, $a^{(l)}$ is the intermediate output, $\odot$ is the element-wise multiplication, and $r^{(l)}$ is the dropout mask sampled from $Bernoulli$ distribution. While dropping nodes from NN, expected loss $\mathbb{E}(.)$ increases which enforces regularization penalty on NN to achieve better generalization (Mianjy, Arora, and Vidal 2018).

Introducing strength parameter

As shown in Figure 3, in the initial few iterations of training with and without dropout, network performance is almost similar. The effectiveness of dropout can be observed after few iterations of training. Dropping some of the trained nodes may lead to more number of active nodes in the network. Hence, the performance of the network can be improved. Utilizing this observation and the discussion presented with respect to Figure 2 (about active/inactive nodes), we hypothesize that a guided dropout can lead to better generalization of a network. The proposed guided dropout utilizes the strength of nodes for generation of the dropout mask. In the proposed formulation, strength is learned by the network itself via Stochastic Gradient Descent (SGD) optimization. Mathematically, it is expressed as:

$$a_j^{(l+1)} = t_j^{(l+1)} \odot max\left(0, w_{j\times i}^{(l+1)} a_i^l + b_j^{(l+1)}\right) \quad (2)$$

where, $\mathbf{t}^{(l)}$ is sampled from $uniform$ distribution (assuming all nodes have equal contribution). It can also be used to measure the importance of the feature-map for CNN networks. Therefore, Equation 2 can be rewritten as:

$$\mathbf{a}^{(l+1)} = \mathbf{t}^{(l+1)} \odot max\left(0, \mathbf{a}^l * \mathbf{W}^{(l+1)} + b^{(l+1)}\right) \quad (3)$$

where, $*$ is a convolution operation, max(0, .) is a RELU operation[1], and $\mathbf{a}^l$ is a three-dimensional feature map (for ease of understanding, subscript has been removed).

Strength parameter in matrix decomposition: To understand the behavior of the proposed strength parameter $\mathbf{t}$ in a simpler model, let the projection of input $x \in \mathbb{R}^{d_2}$ on $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ represent label vector y ($\in \mathbb{R}^{d_1}$). Matrix $\mathbf{W}$ can be linearly compressed using singular value decomposition (SVD), i.e., $W = Udiag(t)V^T$. Here, top few entries of $diag(t)$ can approximate the matrix $\mathbf{W}$ (Denton et al. 2014). This concept can be utilized in the proposed guided dropout.

Strength parameter in two hidden layers of an NN: In an NN environment, let $V \in \mathbb{R}^{d_2 \times r}$ and $U \in \mathbb{R}^{d_1 \times r}$ be the weight matrices of the first and second hidden layers of NN, respectively. The hypothesis class can be represented as $h_{U,V}(x) = UV^T x$ (Mianjy, Arora, and Vidal 2018). In case of *Guided Dropout*, hidden node is parameterized as $h_{U,V,t}(x) = Udiag(t)V^T x$ which is similar to the SVD decomposition where parameter $\mathbf{t}$ is learned via backpropagation. Therefore, t can be considered as a strength of a node which is directly proportional to the contribution of the node in NN performance.

In this case, the weight update rule for parameters U, V and $\mathbf{t}$ on the $(s+1)^{th}$ batch can be written as:

$$U_{s+1} \leftarrow U_s - \eta\left(\frac{1}{\theta}U_s diag(t_s \odot r_s)V_s^T x_s - y_s\right) x_s^T V_s diag(t_s \odot r_s)$$

$$V_{s+1} \leftarrow V_s - \eta x_s\left(\frac{1}{\theta}x_s^T V_s diag(t_s \odot r_s)U_s^T - y_s^T\right) U_s diag(t_s \odot r_s)$$

$$diag(t_{s+1}) \leftarrow diag(t_s) - \eta U_s^T\left(\frac{1}{\theta}U_s diag(t_s \odot r_s)V_s^T x_s - y_s\right) x_s^T V_s \quad (4)$$

where, $\{(x_s,\ y_s)\}_{s=0}^{S-1}$ is the input data, $(1-\theta)$ is the dropout rate, η is the learning rate, and r is the dropout mask. For the initial few iteration, r is initialized with ones. However, after few iterations of training, r is generated using Equation 5.

In the proposed algorithm active nodes are dropped in two ways:

1. Guided Dropout (top-k): Select (top-k) nodes (using strength parameter) to drop
2. Guided Dropout (DR): Drop Randomly from the active region.

Proposed Guided Dropout (top-k): While dropping (top-k) nodes based on the strength, the mask for the proposed guided dropout can be represented as:

$$\mathbf{r}^l = \mathbf{t}^l \leq th,\ where,\ th = \max_{\lfloor N\times(1-\theta)\rfloor} \mathbf{t}^l \quad (5)$$

$\max_{\lfloor N\times(1-\theta)\rfloor}$ is defined as the k large elements of $\mathbf{t}$ where, $(1-\theta)$ is the percentage ratio of nodes needed to be dropped and N is the total number of nodes. The generated mask $\mathbf{r}^l$ is then utilized in equation $\widetilde{\mathbf{y}}^{(l)} = \mathbf{r}^{(l)} \odot \mathbf{y}^{(l)}$ to drop the nodes. Since the number of dropped nodes are dependent on the total number of nodes N and percentage ratio $(1-\theta)$, expected loss can be measured by $\mathbb{E}_{b,x}[||y - \frac{1}{\theta}Udiag(r)V^T x||^2]$. If dropout mask $\mathbf{r}^l$ drops $top-k$ nodes, the expected loss would be maximum with respect to conventional dropping nodes. Therefore, guided dropout ($top-k$) would impose maximum penalty in NN loss.

Proposed Guided Dropout (DR): The second way of generating guided dropout mask is to select the nodes from

[1] In the case of other activation functions, intermediate feature maps might have negative values. Therefore, $|t|$ (mod of 't') can be considered as strength parameter. In this case, 't' value approaching to zero represents low strength and node associated with low strength can be considered as an inactive node.

the active region, i.e., nodes are Dropped Randomly (DR) from the active region only. Since the number of inactive nodes are large and have a similar strength; therefore, to find the active or inactive region, number of elements in all the bins[2] have been computed. The maximum number of elements among all the bins is considered as the count of inactive nodes f_m. Thus, f_m and $(N - f_m)$ are the number of inactive and active nodes, respectively. Here, $(1 - \theta)$ is the probability for sampling dropout mask for active region nodes using $Bernoulli$ distribution. Probability with respect to the total number of nodes should be reduced to maintain the mean μ in the training phase. Therefore, new probability with respect to N is modified as $(1 - \frac{f_m}{N}(1-\theta))$. For the proposed guided dropout (DR), when the nodes are dropped randomly from the active region, in Equation 4, $\frac{1}{\theta}$ will be modified as $\frac{1}{\frac{f_m}{N}(1-\theta)}$. We have carefully mentioned $(1-\theta)$ as the percentage ratio for the proposed guided dropout $(top - k)$. In case of guided dropout $(top - k)$, the generated mask might be fixed until any low strength node can replace the $(top - k)$ nodes. On the other hand, for the proposed guided dropout (DR), $(1 - \theta)$ is the dropout probability.

Why Guided Dropout Should Work?

Dropout improves the generalization of neural networks by preventing co-adaptation of feature detectors. However, it is our assertion that guidance is essential while dropping nodes from the hidden layers. Guidance can be provided based on the regions where nodes are dropped randomly or top few nodes are dropped from the active region. To understand the generalization of the proposed guided dropout, we have utilized Lemma A.1 from (Mianjy, Arora, and Vidal 2018). In their proposed lemma: Let $x \in \mathbb{R}^{d_2}$ be distributed according to distribution D with $\mathbb{E}_x[xx^T] = \mathbf{I}$. Then, for $\mathcal{L}(U, V) := \mathbb{E}_x[||y - UV^T x||^2]$ and $f(U, V) := \mathbb{E}_{b,x}[||y - \frac{1}{\theta}Udiag(r)V^T x||^2]$, it holds that

$$f(U, V) = \mathcal{L}(U, V) + \lambda \sum_{i=1}^{n} ||u_i||^2 ||v_i||^2 \qquad (6)$$

Furthermore, $\mathcal{L}(U, V) = ||W - UV^T||_F^2$, where, $diag(r) \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{d_2\times n}$, $U \in \mathbb{R}^{d_1\times n}$, $W \in \mathbb{R}^{d_1\times d_2}$, $\lambda = \frac{1-\theta}{\theta}$. (The proof of this lemma is given in (Mianjy, Arora, and Vidal 2018)).

According to the above mentioned lemma, it can be observed that the guided dropout assists NN to have a better generalization. Let r be sampled from $Bernoulli$ distribution at every iteration to avoid overfitting in conventional dropout. In guided dropout, mask r' is generated based on the strength value t. For Equation $\mathcal{L}(U, V) = ||W - Udiag(t)V^T||_F^2$, high strength nodes can be chosen to form mask r'. Therefore, loss $\mathbb{E}_{b,x}[||y - \frac{1}{\theta}Udiag(r')V^T x||^2] \geq \mathbb{E}_{b,x}[||y - \frac{1}{\theta}Udiag(r)V^T x||^2]$. In this case, penalty would increase while dropping higher strength nodes. The expected loss would be same only if $r = r'$. If $r \neq r'$, the regularization imposed by the proposed guided dropout increases in the training process. Hence, optimizing the loss in the training process helps inactive nodes to improve their strength in the absence of higher strength nodes.

From Equation 6, the path regularization term $\lambda \sum_{i=1}^{n} ||u_i||^2 ||v_i||^2$ regularizes the weights u_i and v_i of the inactive node such that the increase in loss due to dropping higher strength nodes can be minimized. Thus, the worst case of generalization provided by the proposed guided dropout should be equal to the generalization provided by the conventional dropout.

Implementation Details

Experiments are performed on a workstation with two 1080Ti GPUs under PyTorch (Paszke et al. 2017) programming platform. The program is distributed on both the GPUs. Number of epoch, learning rate, and batch size are kept as 200, $[10^{-2}, ..., 10^{-5}]$, and 64, respectively for all the experiments. Learning rate is started from 10^{-2} and is reduced by a factor of 10 at every 50 epochs. For conventional dropout, the best performing results are obtained at 0.2 dropout probability. In the proposed guided dropout, 40 epochs have been used to train the strength parameter. Once the strength parameter is trained, dropout probabilities for guided dropout (DR) are set to 0.2, 0.15, and 0.1 for 60, 50, and 50 epochs, respectively. However, after strength learning, dropout ratio for guided dropout (top-k) are set to [0.2, 0.0, 0.15, 0.0, 0.1, 0.0] for [10, 40, 10, 40, 10, 50] epochs, respectively.

Experimental Results and Analysis

The proposed method has been evaluated using three experiments: i) guided dropout in neural network, ii) guided dropout in deep network (ResNet18 and Wide ResNet 28-10), and iii) case study with small sample size problem. The databases used for evaluation are MNIST, SVHN, CIFAR10, CIFAR100, and Tiny ImageNet.

The proposed guided dropout is compared with state-of-art methods such as Concrete dropout[3](Gal, Hron, and Kendall 2017), Adaptive dropout (Standout)[4] (Ba and Frey 2013), Variational dropout[5] (Kingma, Salimans, and Welling 2015), and Gaussian dropout[5]. Alpha-dropout (Klambauer et al. 2017) has also been proposed in literature. However, it is specifically designed for SELU activation function. Therefore, to have a fair comparison, results of the alpha-dropout are not included in Tables.

Database and Experimental Protocol

Protocol for complete database: Five benchmark databases including MNIST (LeCun et al. 1998), CIFAR10 (Krizhevsky and Hinton 2009), CIFAR100 (Krizhevsky and Hinton 2009), SVHN (Netzer et al. 2011), and Tiny ImageNet (TinyImageNet 2018) have been used to evaluate the proposed method. MNIST is only used for benchmarking Neural Network with conventional

[2]In this case, 100 equally spaced bins are chosen.

[3]https://tinyurl.com/yb5msqrk

[4]https://tinyurl.com/y8u4kzyq

[5]https://tinyurl.com/y8yf6vmo

Table 1: Test accuracy (%) on CIFAR10 and CIFAR100 (Krizhevsky and Hinton 2009) databases using four different architectures of a three layer Neural Network. (Top two accuracies are in bold).

Algorithm	**CIFAR10**				**CIFAR100**			
	1024, 3	2048, 3	4096, 3	8192, 3	1024, 3	2048, 3	4096, 3	8192, 3
Without Dropout	58.59	59.48	**59.72**	59.27	28.86	30.01	30.73	32.02
With Dropout	**58.77**	59.61	59.62	59.86	**31.52**	**31.63**	**31.37**	31.63
Concrete Dropout	57.38	57.64	57.45	55.28	28.03	29.09	28.91	31.02
Adaptive Dropout	55.05	55.45	56.84	57.01	27.82	28.27	28.62	28.65
Variational Dropout	48.90	52.08	53.48	54.90	17.02	20.64	23.32	24.53
Gaussian Dropout	56.12	56.52	56.94	57.34	27.24	28.34	28.87	29.81
Strength only	58.30	58.92	59.21	59.49	29.66	30.20	30.84	31.12
Proposed Guided Dropout (top-k)	58.75	**59.65**	59.64	**59.92**	30.92	31.59	31.34	**32.11**
Proposed Guided Dropout (DR)	**59.84**	**60.12**	**60.89**	**61.32**	**31.88**	**32.78**	**33.01**	**33.15**

Table 2: Test accuracy (%) on SVHN (Netzer et al. 2011) and Tiny ImageNet (TinyImageNet 2018) databases using four different architectures of a three layer Neural Network (NN). (Top two accuracies are in bold).

Algorithm	**SVHN**				**TinyImageNet**			
	1024, 3	2048, 3	4096, 3	8192, 3	1024, 3	2048, 3	4096, 3	8192, 3
Without Dropout	**86.36**	**86.72**	**86.82**	86.84	12.42	13.74	14.64	15.21
With Dropout	85.98	86.60	86.77	86.79	**16.39**	14.28	14.69	14.44
Concrete Dropout	83.57	84.34	84.97	85.53	11.98	12.50	12.65	14.85
Adaptive Dropout	77.67	79.68	80.89	81.96	12.41	12.98	13.75	14.17
Variational Dropout	74.28	77.91	80.22	81.52	7.95	10.08	12.91	14.69
Gaussian Dropout	72.46	78.07	80.42	80.74	13.88	**15.67**	**15.76**	15.94
Strength only	85.76	85.92	85.91	86.83	12.11	13.52	13.95	14.63
Proposed Guided Dropout (top-k)	86.12	86.57	86.78	**86.85**	15.47	15.45	15.55	**16.01**
Proposed Guided Dropout (DR)	**87.64**	**87.92**	**87.95**	**87.99**	**17.59**	**18.84**	**18.41**	**17.74**

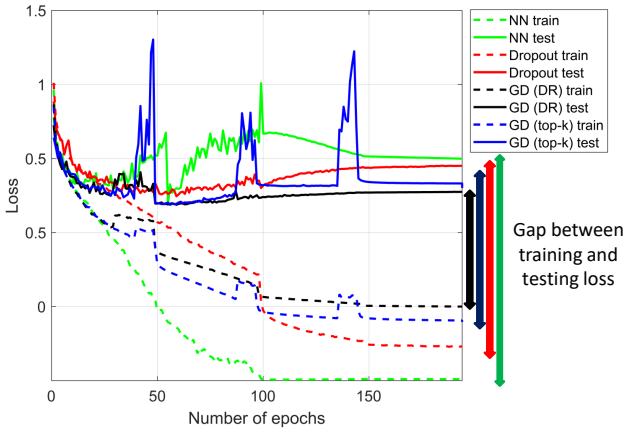

Figure 3: Illustrating of training and testing losses at every epoch. On the CIFAR10 dataset, the proposed method is compared with the conventional dropout method. It can be observed that the gap between training and testing loss is minimum in proposed guided dropout. (Best viewed in color).

dropout (Srivastava et al. 2014). The MNIST dataset contains $70k$ grayscale images pertaining to 10 classes (28×28 resolution). The CIFAR10 dataset contains $60k$ color images belonging to 10 classes (32×32 resolution). The experiments utilize $50k$ training samples and $10k$ as the test samples. CIFAR100 has a similar protocol with 100 classes. The protocol for CIFAR100 also has $50k$ and $10k$ training-testing split. The SVHN dataset contains $73,257$ training samples and $26,032$ testing samples. Tiny ImageNet dataset is a subset of the ImageNet dataset with 200 classes. It has images with 64×64 resolution with $100k$ and $10k$ samples for training and validation sets, respectively. The test-set label is not publicly available. Therefore, validation-set is treated as test-set for all the experiments on Tiny ImageNet.

Protocol for small sample size problem: Recent literature has emphasized the importance of deep learning architecture working effectively with small sample size problems (Keshari et al. 2018). Therefore, the effectiveness of the proposed algorithm is tested for small sample size problem as well. The experiments are performed on Tiny ImageNet database with three-fold cross validation. From the entire training set, $200, 400..., 1k, 2k, .., 5k$ samples are randomly chosen to train the network and evaluation is performed on the validation set.

Evaluation of Guided Dropout in Dense Neural Network (NN) Architecture

To showcase the generalization of the proposed method, training and testing loss at every epoch is shown in Figure 3. A three layer NN with 8192 nodes at each layer is trained without dropout, with dropout, and with the two proposed

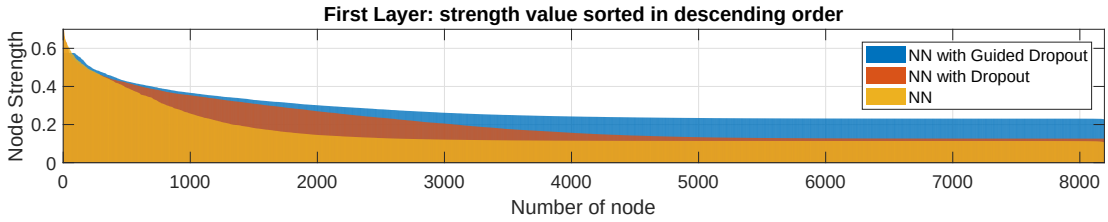

Figure 4: Illustrating the learned strength values of the first hidden layer nodes for the CIFAR10 database with NN$[8192, 3]$. Strength of nodes is improved by utilizing the proposed guided dropout in comparison to with/without conventional dropout. (Best viewed in color).

Table 3: Test accuracy (%) on the MNIST (LeCun et al. 1998) database using three layer Neural Network (NN). (Top two accuracies are in bold).

Algorithm	**Number of nodes, Layers**			
	1024, 3	**2048, 3**	**4096, 3**	**8192, 3**
Without Dropout	98.44	98.49	98.42	98.41
With Dropout	98.45	**98.67**	98.50	98.53
Concrete Dropout	**98.66**	98.60	**98.62**	98.59
Adaptive Dropout	98.31	98.33	98.34	98.40
Variational Dropout	98.47	98.55	98.58	98.52
Gaussian Dropout	98.35	98.43	98.47	98.44
Strength only	98.42	98.51	98.40	98.46
Proposed Guided Dropout (top-k)	98.52	98.59	98.61	**98.68**
Proposed Guided Dropout (DR)	**98.93**	**98.82**	**98.86**	**98.89**

guided dropout algorithms $top - k$, and DR. It can be inferred that the proposed guided dropout approaches help to reduce the gap between the training and testing losses.

The proposed guided dropout is evaluated on three layer Neural Network (NN) with four different architectures as suggested in (Srivastava et al. 2014). Tables 1 to 3 summarize test accuracies (%) on CIFAR10, CIFAR100, SVHN, Tiny ImageNet, and MNIST databases. It can be observed that the proposed guided dropout (DR) performs better than existing dropout methods. In large parameter setting such as three layer NN with 8192 nodes, the proposed guided dropout (top-k) algorithm also shows comparable performance. For NN$[1024, 3]$ architecture, conventional dropout is the second best performing algorithm on CIFAR10, CIFAR100, and Tiny ImageNet databases.

We have claimed that the strength parameter is an essential element in NN to measure the importance of nodes. Though the number of training parameters are increased but this overhead is less than 0.2% of the total number of parameters of a NN[6].

Figure 4 represents the learned strength value of the first hidden layer of NN$[8196, 3]$. It can be observed that the conventional dropout improves the strength of hidden layer nodes. However, the strengths are further improved upon by utilizing the proposed guided dropout.

[6]For a NN with three hidden layers of 8192 nodes each, total number of learning parameters is only $8192 \times 8192 + 8192 \times 8192 = 134,217,728$ and overhead of strength parameter is $24,576$.

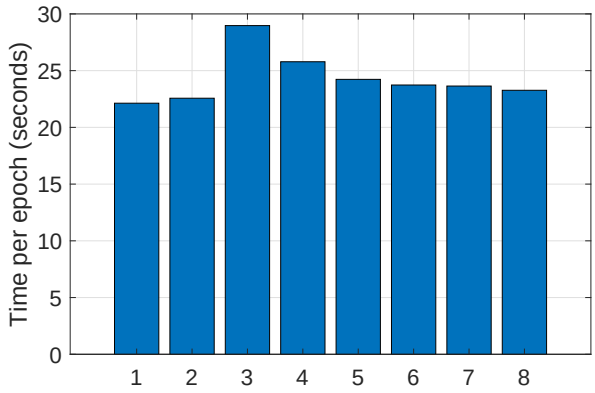

Figure 5: Illustration of time taken in per epoch. X-axis represents without dropout (1), with dropout (2), concrete dropout (3), adaptive dropout (4), variational dropout (5), Gaussian dropout (6), proposed guided dropout (top-k) (7), and proposed guided dropout (DR) (8) algorithms, respectively.

For understanding the computational requirements, a NN$[8192, 3]$ has been trained and time taken without dropout, with dropout, concrete dropout, adaptive dropout, variational dropout, Gaussian dropout, proposed guided dropout (top-k), and proposed guided dropout (DR) and the results are reported for one epoch. Figure 5 summarizes the time (in seconds) for these variations, which clearly shows that applying the proposed dropout approach does not increase the time requirement.

Evaluation of Guided Dropout in Convolutional Neural Network (CNN) Frameworks

The proposed guided dropout is also evaluated on CNN architectures of ResNet18 and Wide-ResNet 28-10. On the same protocol, the proposed guided dropout performance is compared with existing state-of-the-art dropout methods. Table 4 summarizes test accuracies of four bench marking databases. It can be observed that on CIFAR10 (C10), dropout is providing second best performance after the proposed algorithm. On CIFAR100 (C100), without dropout is

Table 4: Test accuracy (%) on CIFAR10, CIFAR100 (Krizhevsky and Hinton 2009) (in Table written as C10, C100), SVHN (Netzer et al. 2011), and Tiny ImageNet (TinyImageNet 2018) databases using CNN architectures of ResNet18 and Wide-ResNet 28-10. (Top two accuracies are in bold).

Algorithm	ResNet18				Wide-ResNet 28-10			
	C10	C100	SVHN	Tiny ImageNet	C10	C100	SVHN	Tiny ImageNet
Without Dropout	93.78	**77.01**	96.42	61.96	96.21	81.02	96.35	63.57
With Dropout	**94.09**	75.44	**96.66**	**64.13**	**96.27**	**82.49**	**96.75**	**64.38**
Concrete Dropout	91.33	74.74	92.63	62.95	92.63	75.94	92.79	–
Adaptive Dropout	90.45	73.26	92.33	61.14	79.04	52.12	90.40	62.15
Variational Dropout	94.01	76.23	96.12	62.75	96.16	80.78	96.68	64.36
Gaussian Dropout	92.34	75.11	95.84	60.33	95.34	79.76	96.02	63.64
Strength only	93.75	76.23	96.34	62.06	95.93	80.79	96.31	64.13
Proposed Guided Dropout (top-k)	94.02	76.98	96.62	64.11	96.22	82.31	96.42	64.32
Proposed Guided Dropout (DR)	**94.12**	**77.52**	**97.18**	**64.33**	**96.89**	**82.84**	**97.23**	**66.02**

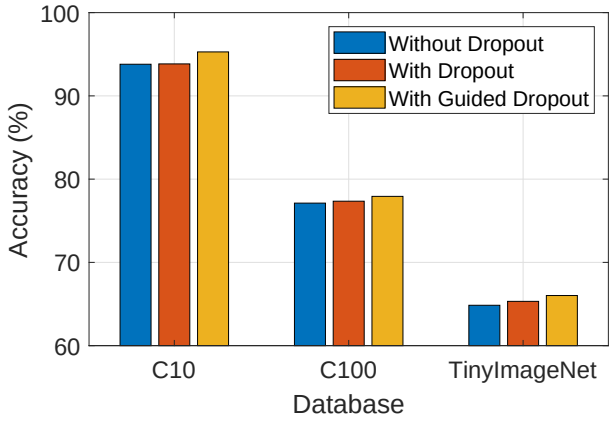

Figure 6: Classification accuracies on C10, C100, and Tiny ImageNet databases. The performance is measured with ResNet152 CNN architecture without dropout, with traditional dropout, and with the proposed guided dropout. (Best viewed in color).

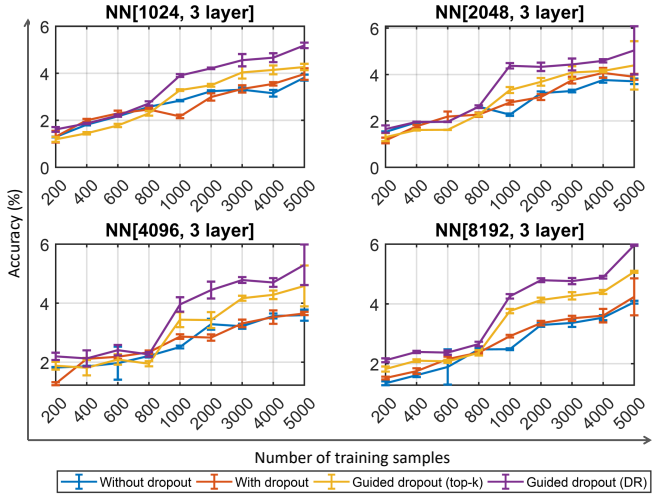

Figure 7: Results of small sample size experiments: Accuracies on varying training samples of the Tiny ImageNet dataset. The performance has been measured on four different dense NN architectures. (Best viewed in color).

providing second best performance and the proposed guided dropout (DR) is providing the best performance. In case of Wide-ResNet 28-10 which has larger parameter space than ResNet18, conventional dropout consistently performs second best after the proposed guided dropout (DR). It improves the Wide-ResNet 28-10 network performance by 0.62%, 0.35%, 0.48%, and 1.64% on C10, C100, SVHN, and Tiny ImageNet databases, respectively. We have also computed the results using ResNet152 CNN architecture on C10, C100, and Tiny ImageNet databases. As shown in Figure 6, even with a deeper CNN architecture, the proposed guided dropout performs better than the conventional dropout method.

Small Sample Size Problem

Avoiding overfitting for small sample size problems is a challenging task. A deep neural network, which has a large number of parameters, can easily overfit on small size data. For measuring the generalization performance of models, (Bousquet and Elisseeff 2002) suggested to measure the generalization error of the model by reducing the size of the training dataset. Therefore, we have performed three fold validation along with varying the size of the training data.

The experiments are performed with ResNet-18 and four dense neural network architectures. As shown in Figures 7 and 8, with varying training samples of the Tiny ImageNet database, the proposed guided dropout (DR) yields higher accuracies compared to the conventional dropout.

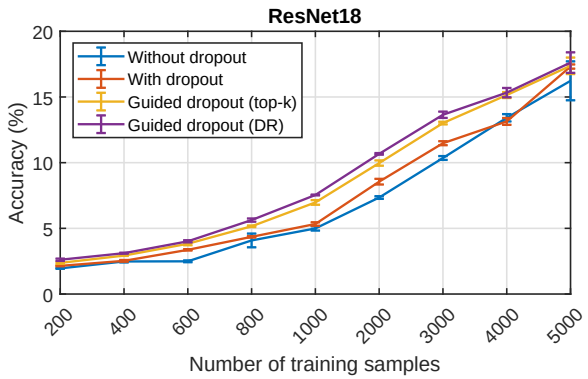

Figure 8: Classification accuracies obtained with varying the training samples for the Tiny ImageNet dataset. The performance is measured with ResNet18 CNN architecture. (Best viewed in color).

Discussion and Conclusion

Dropout is a widely used regularizer to improve the generalization of neural network. In the dropout based training, a mask is sampled from $Bernoulli$ distribution with $(1-\theta)$ probability which is used to randomly drop nodes at every iteration. In this research, we propose a guidance based dropout, termed as guided dropout, which drops active nodes with high strength in each iteration, as to force non-active or low strength nodes to learn discriminative features. During training, in order to minimize the loss, low strength nodes start contributing in the learning process and eventually their strength is improved. The proposed guided dropout has been evaluated using dense neural network architectures and convolutional neural networks. All the experiments utilize benchmark databases and the results showcase the effectiveness of the proposed guided dropout.

Acknowledgement

R. Keshari is partially supported by Visvesvaraya Ph.D. fellowship. R. Singh and M. Vatsa are partly supported by the Infosys Center of Artificial Intelligence, IIIT Delhi, India.

References

Ba, J., and Frey, B. 2013. Adaptive dropout for training deep neural networks. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *NIPS*. Curran Associates, Inc. 3084–3092.

Bousquet, O., and Elisseeff, A. 2002. Stability and generalization. *JMLR* 2(Mar):499–526.

Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 1269–1277.

Gal, Y., and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 1050–1059.

Gal, Y.; Hron, J.; and Kendall, A. 2017. Concrete dropout. In *NIPS*, 3584–3593.

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

Hinton, G. E.; Osindero, S.; and Teh, Y.-W. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18(7):1527–1554.

Keshari, R.; Vatsa, M.; Singh, R.; and Noore, A. 2018. Learning structure and strength of CNN filters for small sample size training. In *CVPR*, 9349–9358.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. In *NIPS*, 2575–2583.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-normalizing neural networks. In *NIPS*, 971–980.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Technical report*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Mianjy, P.; Arora, R.; and Vidal, R. 2018. On the implicit bias of dropout. *arXiv preprint arXiv:1806.09777*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS-W*, volume 2011, 5.

Nowlan, S. J., and Hinton, G. E. 1992. Simplifying neural networks by soft weight-sharing. *Neural computation* 4(4):473–493.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

TinyImageNet. 2018. Tiny ImageNet tiny imagenet visual recognition challenge. `https://tiny-imagenet.herokuapp.com/`. Accessed: 14th-May-2018.

Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *ICML*, 1058–1066.

Wang, S., and Manning, C. 2013. Fast dropout training. In *ICML*, 118–126.