Neurocomputing 357 (2019) 177–187

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Rademacher dropout: An adaptive dropout for deep neural network via optimizing generalization gap

Haotian Wang[a], Wenjing Yang[a,*], Zhenyu Zhao[b], Tingjin Luo[b], Ji Wang[a], Yuhua Tang[a]

[a] State Key Laboratory of High Performance Computing (HPCL), College of Computer, National University of Defense Technology, PR China
[b] College of Liberal Arts and Sciences, National University of Defense Technology, PR China

ARTICLE INFO

Article history:
Received 21 September 2018
Revised 19 February 2019
Accepted 4 May 2019
Available online 11 May 2019

Communicated by Dr Wen Wujie

Keywords:
Overfitting
Dropout
Rademacher complexity
Generalization gap
Deep neural network

ABSTRACT

Dropout plays an important role in improving the generalization ability in deep learning. However, the empirical and fixed choice of dropout rates in traditional dropout strategies may increase the generalization gap, which is counter to one of the principle aims of dropout. To handle this problem, in this paper, we propose a novel dropout method. By the theoretical analysis of Dropout Rademacher Complexity, we first prove that the generalization gap of a deep model is bounded by a constraint function related to dropout rates. Meanwhile, we derive a closed form solution via optimizing the constraint function, which is a distribution estimation of dropout rates. Based on the closed form solution, a lightweight complexity algorithm called **Rademacher Dropout** (RadDropout) is presented to achieve the adaptive adjustment of dropout rates. Moreover, as a verification of the effectiveness of our proposed method, the extensive experimental results on benchmark datasets show that RadDropout achieves improvement of both convergence rate and prediction accuracy.

1. Introduction

Deep learning has achieved great success in a number of domains, such as the image processing [1,2], text analysis [3], and control fields [4]. However, the excessive feature learning by deep neural networks (DNNs) might lead to the overfitting phenomenon, which reduces the generalization ability of deep models. Motivated by Kerkar et al. [5,6], the generalization ability of DNNs can be quantitatively measured by the generalization gap, which is the difference between the empirical risk (training error) and the expected risk (generalization error). Moreover, the generalization gap of DNNs [7] can be formulated as

$$\text{the generalization gap} \triangleq |R_{\text{exp}} - R_{\text{emp}}|, \tag{1}$$

where R_{exp} and R_{emp} represent the expected risk and the empirical risk of deep models, respectively. Once a deep model is overfitting, R_{exp} will be much larger than R_{emp} and its generalization gap is written as

$$\text{the generalization gap} = R_{\text{exp}} - R_{\text{emp}}. \tag{2}$$

To enhance the generalization ability of DNNs, various techniques have been developed, such as ensemble learning [8], batch normalization (BN) [9], and dropout [10,11]. Ensemble learning prevents overfitting via a combination of multiple classifiers [12]. BN was proposed to reduce the internal covariate shift through the normalization operations on the batches of each layer. However, ensemble learning requires expensive computational resources and the performance of BN depends on the batch-size [9]. To improve the generalization ability of DNNs stably and efficiently, in this paper, we focus on studying dropout for its simplicity and remarkable effectiveness. Dropout is one of the most widely adopted regularization approaches in deep learning [13–16]. The strategy of dropout is to randomly drop neurons within one layer while training DNNs [10]. Then, the generalization ability of a deep model is improved by breaking the fixed combination of features.

Despite its empirical success, current theoretical analysis of the dropout technique remains rudimentary and vague [17]. This can be explained by the fact that the fundamental theory of the DNN still remains a riddle, which is called ‘Black Box’ [18]. Therefore, the effectiveness of dropout mechanism can only be estimated from some existing techniques, such as Bayesian theory [19], optimization analysis [20], and statistic generalization bound [21,22]. For linear models, dropout training originally was analyzed as ensemble learning in shallow networks [11]. Moreover, Warde-Farley et al. [16] verified the effectiveness of the geometric average approximation, which combines the training results from multiple sub-networks. Motivated by norm regularization theory, Wager et al. [23] analyzed, for generalized linear models, that dropout is the first-order equivalent to an ℓ_2 regularizer. For deep

* Corresponding author.
E-mail address: wenjing.yang@nudt.edu.cn (W. Yang).

https://doi.org/10.1016/j.neucom.2019.05.008
0925-2312/© 2019 Elsevier B.V. All rights reserved.

models, Gal et al. [19,24] proposed probabilistic interpretations for drop training, which proved that DNNs trained with dropout are mathematically equivalent to an approximation to a well-known Bayesian model. Recently, a battery of studies has emerged that attempt to explain dropout training by risk bound analysis and Rademacher Complexity [25]. Unlike those data-independent complexities, Rademacher Complexity can attain a much more compact generalization representation [26]. For DNNs trained with dropout, Mou et al. [17] derived that the generalization error is bounded by the sum of two offset Rademacher Complexities. Gao et al. [21] developed Rademacher Complexity into Dropout Rademacher Complexity and obtained a compact estimation of the expected risk.

Although the theoretical analysis is still vague, empirical experiments show that the effect of dropout is intrinsically related to the choice of dropout rates [20]. For convenience, the traditional dropout method made an assumption that the distribution of dropout rates obeys the Binomial Bernoulli distribution. Based on this viewpoint, traditional dropout decides the value of dropout rates empirically, which sets the value of dropout rates by some rule-of-thumb [10,11]. Meanwhile, the traditional dropout method treats each neuron in one layer equally, where each neuron shares the same dropout rates. However, different neurons represent different features and they contribute to the prediction to different extents [20,27]. Combining the aforementioned insights, there is still room to improve traditional dropout method by adaptively choosing dropout rates for DNNs.

To improve the generalization performance of DNNs, a number of variants of dropout are proposed to design adaptive mechanisms for the update of dropout rates. From the probabilistic viewpoint, the variants of dropout mainly concentrated on studying the distribution of dropout rates. Some researchers assumed that the dropout rates obey the specific distribution as a prior [10,20,27]. For example, Ba et al. [27] held that dropout rates obey the binomial Bernoulli distribution and constructed a binary belief network over the DNNs, which generates the dropout rates adaptively by minimizing the energy function. However, the additional binary belief network will result in more computational overhead when the model size increases. Moreover, Li et al. [20] sampled dropout rates from a prior multinomial distribution and proposed an evolutionary dropout via risk bound analysis for optimization of a Stochastic Gradient Descent (SGD) learning algorithm [28]. Aside from assuming the prior distribution, another category of dropout variants attempts to estimate the distribution of dropout rates via some optimization framework [17,19,22,24,29,30]. Based on the Bayesian optimization framework, a variety of dropout methods were proposed [19,24,30,31]. Based on deep Bayesian learning, Gal et al. [29] proposed "concrete dropout" to give improved performance and better calibrated uncertainties. Through variational Bayesian inference theory, Kingma et al. [30] explored an extended version of Gaussion dropout called "variational dropout" with local re-parameterization. Recently, an increasing number of studies have tried to estimate the distribution of dropout rates via risk bound optimization [17,22]. Through Rademacher Complexity, Zhai et al. [22] proposed an adaptive dropout regularizer on the objective function. It is worth noting that our research is fundamentally different from their work, in that they take the term of Rademacher Complexity as the regularizer on the objective function of DNNs and we utilize Rademacher Complexity to estimate and optimize the generalization gap.

In this paper, we propose a novel method to achieve the adaptive adjustment of dropout rates with low computational complexity. In fact, estimating the distribution of dropout rates directly is a challenging task as a grid search over a large amount of hyper parameters [22]. As an alternative, we attempt to estimate the distribution of dropout rates via optimizing the generalization gap. Inspired by the estimation of Gao et al. [21], we first prove that the generalization gap of DNNs trained with dropout is bounded by a constraint function related to dropout rates. Subsequently, to optimize the generalization gap, we minimize the constraint function by a theoretical derivation. As a result, we obtain a closed-form solution in the optimization process, in which the solution represents a distribution estimation of the dropout rates. This solution provides an efficient and concise approach to calculate dropout rates by the batch inputs of the dropout layer. Finally, we propose an adaptive algorithm called Rademacher Dropout (RadDropout) based on the closed-form solution. The algorithm generates the dropout rates adaptively during feed-forward propagation and requires only lightweight complexity. To further justify our method, we conduct several experiments on five benchmark datasets: MNIST, SVHN, NORB, Cifar-100, and TinyImagenet. The scope of the experiments includes a comparison of several traditional and state-of-art dropout approaches. The experimental results illustrate that RadDropout improves both on convergence rate and prediction accuracy.

The main contributions of this paper as follows:

- We first prove that the generalization gap of DNNs trained with dropout is bounded by a constraint function related to dropout rates.
- We optimize the generalization gap by a theoretical derivation. Meanwhile, we obtain a closed-form solution as an distribution estimation of dropout rates.
- We propose RadDropout as a novel dropout algorithm based on our solution, which achieves adaptive adjustment of dropout rates with lightweight complexity.

The reminder of this paper is organized as follows. In Section 2, we present some preliminaries. In Section 3, we detail theoretical derivations and the proposed adaptive algorithm, RadDropout. We present the experimental results on five datasets in Section 4 and draw conclusions in Section 5.

2. Preliminaries

2.1. Expression of DNN

Here, we give the expression of the structure of a fully connected network:

- The deep network is composed of L hidden layers: $\{h_1, h_2, \ldots, h_L\}$ and layer h_i has N_i neurons ($N_{L+1} = 1$).
- The weights between layer h_i and layer h_{i+1} are $\mathbf{W}^i$, where $\mathbf{W}^i \in \mathscr{R}^{N_i \times N_{i+1}}$ $(1 \leq i \leq L)$. Moreover, $\mathbf{W}^i = \{\mathbf{W}^i_1, \mathbf{W}^i_2, \ldots, \mathbf{W}^i_{N_{i+1}}\}$, where $\mathbf{W}^i_j \in \mathscr{R}^{N_i \times 1}$ $(1 <= j <= N_i)$.
- The dropout operation can be considered as a element-wise product of the input vector and the bitmask vector, which is generated by the dropout rates. Following Warde-Farley et al. [16], we use dropout masks to denote to the bitmask vector of the dropout operation in this paper. The dropout masks and related dropout rates of layer h_i are defined as $\mathbf{q}^i = \{q^i_1, \ldots, q^i_{N_i}\}$ and $\mathbf{p}^i = \{p^i_1, \ldots, p^i_{N_i}\}$. Here, q^i_j and p^i_j represent the dropout mask and dropout rate of neuron j in layer h_i, respectively.
- The input of the network is defined as $\mathbf{X} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\}$ and the output of the network is defined as $F(\mathbf{X}, \mathbf{W}, \mathbf{q})$, where n is the number of whole samples. For the input of each layer, $\mathbf{\Phi}^i_k$ is the k-th sample's input of layer h_i, where $\mathbf{\Phi}^i_k = \{\Phi^i_k(1), \Phi^i_k(2), \ldots, \Phi^i_k(N_i)\}$. Note that $\mathbf{\Phi}^0_k = \mathbf{x_k}$.
- The activation function σ is a P-Lipschitz function throughout this study, which includes the most commonly used activation functions, e.g., *sigmoid* [32], *tanh* [33] and *relu* [34]. Meanwhile, l is the loss function. With bounded $F(\mathbf{X}, \mathbf{W}, \mathbf{q})$ and y, loss functions, including entropy loss and square loss, are both Lipschitz

functions with the first arguments [21]. Different from the activation function, we suppose that l is a D-Lipschitz function with its first argument. In addition, we detail the definition of Lipschitz function in Section 2.3.

For the convenience of later analysis, the expression of our L-layer network is expressed in recursive form as in [21]. Therefore the output of the network for the jth sample is defined as $F(\mathbf{X}, \mathbf{W}, \mathbf{q})_j$:

$$F(\mathbf{X}, \mathbf{W}, \mathbf{q})_j = \mathbf{W}^L \cdot \mathbf{\Phi}_k^L, \tag{3}$$

just as the kth sample's input of h_i, $\mathbf{\Phi}_k^i$ can be written as

$$\mathbf{\Phi}_k^i = \{\sigma(\mathbf{W}_1^{i-1} \cdot (\mathbf{\Phi}_k^{i-1} \circ \mathbf{q}^{i-1}))\ \dots,\ \sigma(\mathbf{W}_{N_{i-1}}^{i-1} \cdot (\mathbf{\Phi}_k^{i-1} \circ \mathbf{q}^{i-1}))\}, \tag{4}$$

where $\circ$ is an element-wise product operation.

2.2. Dropout training

Dropout can be considered a kind of noise adding to input during training phase [11]. When we add dropout layer after the $i-th$ layer h_i with K_i neurons, the dropout operation can be written as

$$F_{\text{dropout}}(\mathbf{\Phi}_k^i) = \mathbf{q}^i \circ \mathbf{\Phi}_k^i. \tag{5}$$

With the drop rate p, each element of the mask vector in $hidden_i$ follows the distribution as follows:

$$q_j^i \sim \hat{B}(p_j^i) \quad (1 \le j \le N_i). \tag{6}$$

Here, a factor $\frac{1}{1-p}$ is multiplied with dropout mask and we use $\hat{B}$ to refer the "scaled Bernoulli" distribution in [11]. Thus, the distribution of $\mathbf{q}^i$ follows that for $1 \le j \le N_i$, $Prob(q_j^i = 0) = p_j^i$ and $Prob(q_j^i = \frac{1}{1-p_j^i}) = 1 - p_j^i$. Using a scaled dropout mask can ensure that $E_\mathbf{q}[\mathbf{\Phi}^i] = \mathbf{\Phi}^i$; thus, we do not need to perform an extra scaling operation for dropout training [11,20].

2.3. Dropout Rademacher complexity

As the data-independent complexity, Rademacher complexity can attain a much more compact generalization representation [35]. Based on Rademacher Complexity, Gao et al. [21] proposed Dropout Rademacher Complexity and provided a powerful theoretical tool for our analysis. Supposing that $\mathscr{H}$ is a real-valued function space, we now list the definition of Dropout Radmeacher Complexity and related lemmas as follows:

Definition 1 (Lispschitz Function). In particular, a real-valued function $f: \mathscr{R} \to \mathscr{R}$ is called K-Lipschitz continuous if there exists a positive real constant K such that, for all real x_1 and x_2,

$$|f(x_1) - f(x_2)| \le K|x_1 - x_2|. \tag{7}$$

Moreover, a real-valued two variable function $f(x, t)$ is called K-Lipschitz continuous with its first argument if there exists a positive real constant K such that, for all x_1 and x_2,

$$|f(x_1, t) - f(x_2, t)| \le K\|x_1 - x_2\|. \tag{8}$$

Definition 2 (Dropout Rademacher Complexity).

$$\hat{\mathfrak{R}}_n(\mathscr{H}, \mathbf{X}, \mathbf{q}) = E_\epsilon\left[\sup_{h\in\mathscr{H}} \frac{1}{n}\sum_{k=1}^{n} \epsilon_k h(\mathbf{x_k}, \mathbf{q})\right], \tag{9}$$

$$\mathfrak{R}_n(\mathscr{H}) = E_{\mathbf{x},\mathbf{q}}[\hat{\mathfrak{R}}_n(\mathscr{H}, \mathbf{X}, \mathbf{q})], \tag{10}$$

where (9) and (10) are, respectively, called Empirical Dropout Rademacher Complexity and Dropout Rademacher Complexity. Moreover, $\{\epsilon_1, \dots, \epsilon_n\}$ are referred to as Rademacher variables [21].

Lemma 1. *Based on the function space $\mathscr{H}$, we define* $\text{abs}(\mathscr{H}) = \{\Sigma\alpha_i h_i : h_i \in \mathscr{H}\}$, *where* $\Sigma\alpha_i = 1$.

$$\hat{\mathfrak{R}}_n(\mathscr{H}, \mathbf{x}, \mathbf{q}) = \hat{\mathfrak{R}}_n(\text{abs}(\mathscr{H}), \mathbf{x}, \mathbf{q}). \tag{11}$$

Lemma 2. *Let $\mathscr{H}$ be a bounded real-valued function space from some space $\mathscr{Z}$ and $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n \in \mathscr{Z}$. Let $\sigma: \mathscr{R} \to \mathscr{R}$ be a Lipschitz function with constant P and $\sigma(0) = 0$. Then we have*

$$E_\epsilon\left[\sup_{h\in\mathscr{H}} \frac{1}{n}\sum_{k\in[n]} \epsilon_k \sigma(h(\mathbf{z}_k))\right] \le PE_\epsilon\left[\sup_{h\in\mathscr{H}} \frac{1}{n}\sum_{k\in[n]} \epsilon_k h(\mathbf{z}_k)\right]. \tag{12}$$

Lemma 3. *If $l(.,.)$ is a D-Lipschitz function with its first argument, we have*

$$\mathfrak{R}_n(l(F(\mathbf{X}, \mathbf{W}, \mathbf{q}), \mathbf{y}))) \le D\mathfrak{R}_n(F(\mathbf{X}, \mathbf{W}, \mathbf{q})). \tag{13}$$

Lemma 4. *(**Estimation of generalization gap**) We define the empirical risk and expected risk for dropout as R_{emp} and R_{exp}, respectively. Additionally, the loss function l is bounded with constant B and all training samples are i.d.d sampled. Then, the following inequality holds with probability $1 - \delta$:*

$$R_{\text{exp}} - R_{\text{emp}} \le 2\mathfrak{R}_n(l(F(\mathbf{X}, \mathbf{W}, \mathbf{q}), \mathbf{y})) + B\sqrt{\ln(2/\delta)/n}, \tag{14}$$

where $R_{\text{exp}} - R_{\text{emp}}$ is the generalization gap defined in (2).

Remark. The proof and detailed analysis of formulas and lemmas above are given in [21]. Here, we only utilize them to further analyze the generalization gap.

3. Rademacher dropout: optimization of generalization gap

Before further analyzing the generalization gap, we emphasize a supposition as a prior. For the convenience of derivation, here we make a supposition that only one hidden layer h_s has a dropout operation. Since the expression of DNN is in recursive form, the situation of dropout operation for one hidden layer can be easily popularized to the multi-layer dropout.

3.1. Generalization gap constraint function

Improving the generalization ability of DNNs is the fundamental purpose of the dropout strategy. However, the generalization ability is too difficult to directly estimate without the real data distribution. Therefore, the generalization gap is defined as a quantitative measurement, which formalizes the difference between the empirical risk and expected risk. In the research of Gao et al. [21], the generalization gap is proved to be bounded by Dropout Rademacher Complexity in (14). Based on their estimation, we prove that the generalization gap is further bounded by a constraint function related to dropout rates $\mathbf{q}$ in Theorem 1.

Theorem 1. *Let* $\mathbf{W} = \{(\mathbf{W}^1, \mathbf{W}^2, \dots, \mathbf{W}^L)\}$, *and* $\mathbf{W}^i = \{\mathbf{W}_1^i, \mathbf{W}_2^i, \dots, \mathbf{W}_{N_{i+1}}^i\}$. *Supposing that the 2-norm of every column of $\mathbf{W}^s$ is bounded by a constant B_s, we denote $\max_j \|\mathbf{W}_j^s\|_2 \le B_s$. Moreover, the 1-norm of every column of $\mathbf{W}^i (1 \le i \le L-1)$ is bounded by a constant C_i, and we denote $\max_j \|\mathbf{W}_j^i\|_1 \le C_i$. For layer h_L, let $\|\mathbf{W}^L\|_1 \le C_L$. Note that $\mathbf{W}^L$ is a vector here. Then, we have*

$$\mathfrak{R}_n(l(F(\mathbf{X}, \mathbf{W}, \mathbf{q}), \mathbf{y})) \le U(q), \tag{15}$$

where $U(q)$ is a constraint function:

$$U(q) = DB_s\frac{P^{L-s}}{n}\prod_{m=s+1}^{L} C_m E_\mathbf{X}\left[\sqrt{E_\mathbf{q}\left\|\sum_{k=1}^{n} \mathbf{q}^s \circ \mathbf{\Phi}_k^s\right\|_2^2}\right]. \tag{16}$$

Proof. Instead of $\hat{\mathfrak{R}}_n(l(F(\mathbf{X}, \mathbf{W}, \mathbf{q}), \mathbf{y}))$ and $\mathfrak{R}_n(l(F(\mathbf{X}, \mathbf{W}, \mathbf{q}), \mathbf{y}))$, here we use $\hat{K}(q)$ and $K(q)$ to express empirical Dropout

Rademachder Complexity and Dropout Rademacher Complexity, respectively. First, we estimate empirical Dropout Rademacher Complexity $\hat{K}(q)$. Based on Lemma 3 in (13), we have

$$\hat{K}(q) = \hat{\mathfrak{R}}_n(l(F(\mathbf{X},\mathbf{W},\mathbf{q}),\mathbf{y})) \le D\hat{\mathfrak{R}}_n(F(\mathbf{X},\mathbf{W},\mathbf{q})).$$

According to the definition of the empirical Dropout Rademachder Complexity in (9) and the expression of the DNN in (3) and (4), we have

$$\begin{aligned}\hat{K}(q) &\le D\hat{\mathfrak{R}}_n(F(\mathbf{X},\mathbf{W},\mathbf{q}))\\ &= \frac{D}{n}E_\epsilon\left[\sup_{\mathbf{W}^L}\mathbf{W}^L\cdot\left(\sum_{k=1}^n \epsilon_k\mathbf{q}^L\circ\mathbf{\Phi}_k^L\right)\right]\\ &\le \frac{DC_L}{n}E_\epsilon\left[\sup_{\mathbf{W}^L}\frac{\mathbf{W}^L}{\|\mathbf{W}^L\|_1}\cdot\left(\sum_{k=1}^n \epsilon_k\mathbf{q}^L\circ\mathbf{\Phi}_k^L\right)\right].\end{aligned}$$

Based on Lemma 1 in (11), we consider $\frac{W_i^L}{\|\mathbf{W}^L\|_1}$ as α_i and obtain inequalities as follows:

$$\begin{aligned}\hat{K}(q) &\le \frac{DC_L}{n}E_\epsilon\left[\sup_{\mathbf{W}_1^{L-1}}\sum_{k=1}^n \epsilon_k q_1^L\sigma\left(\mathbf{W}_1^{L-1}\cdot\left(\mathbf{q}^{L-1}\circ\mathbf{\Phi}_k^{L-1}\right)\right)\right]\\ &= \frac{DC_L}{n}E_\epsilon\left[\sup_{\mathbf{W}_1^{L-1}}\sum_{k=1}^n \epsilon_k\sigma\left(\mathbf{W}_1^{L-1}\cdot\left(\mathbf{q}^{L-1}\circ q_1^L\mathbf{\Phi}_k^{L-1}\right)\right)\right].\end{aligned}$$

Based on Lemma 2 in (12), we then have

$$\begin{aligned}\hat{K}(q) &\le P\frac{DC_L}{n}E_\epsilon\left[\sup_{\mathbf{W}_1^{L-1}}\sum_{k=1}^n \epsilon_k\left(\mathbf{W}_1^{L-1}\cdot\left(\mathbf{q}^{L-1}\circ q_1^L\mathbf{\Phi}_k^{L-1}\right)\right)\right]\\ &= DPC_LE_\epsilon\left[\sup_{\mathbf{W}_1^{L-1}}\mathbf{W}_1^{L-1}\cdot\left(\frac{1}{n}\sum_{k=1}^n \epsilon_k\mathbf{q}^{L-1}\circ q_1^L\mathbf{\Phi}_k^{L-1}\right)\right].\end{aligned}$$

Similarly, we can obtain the following inequalities in a similar approach as before:

$$\begin{aligned}\hat{K}(q) &\le DP^2C_LC_{L-1}E_\epsilon\left[\sup_{\mathbf{W}_1^{L-2}}\mathbf{W}_1^{L-2}\cdot\left(\frac{1}{n}\sum_{k=1}^n \epsilon_k\mathbf{q}^{L-2}\circ q_1^L q_1^{L-1}\mathbf{\Phi}_k^{L-2}\right)\right]\\ &\le D\frac{P^{L-s}}{n}\prod_{m=s+1}^{L}C_mE_\epsilon\left[\sup_{\mathbf{W}_1^{s}}\mathbf{W}_1^{s}\cdot\left(\sum_{k=1}^n \epsilon_k\mathbf{q}^{s}\circ \prod_{r=s+1}^{L}q_1^r\mathbf{\Phi}_k^{s}\right)\right].\end{aligned}$$

From the Cauchy–Buniakowsky–Schwarz inequality, we have

$$\hat{K}(q) \le DB_s\frac{P^{L-s}}{n}\prod_{m=s+1}^{L}C_mE_\epsilon\left\|\sum_{k=1}^n \epsilon_k\mathbf{q}^{s}\circ \prod_{r=s+1}^{L}q_1^r\mathbf{\Phi}_k^{s}\right\|_2.$$

Combined with the property of Rademacher variables that $E_{\epsilon_{k_1}\epsilon_{k_2}}\epsilon_{k_1}\epsilon_{k_2}=0$ *for* $k_1\neq k_2$ *and* $E_{\epsilon_k\epsilon_k}\epsilon_k\epsilon_k=1$ in [21], the term $E(q)=E_\epsilon\left\|\sum_{k=1}^n\epsilon_k\mathbf{q}^s\circ\prod_{r=s+1}^L q_1^r\mathbf{\Phi}_k^s\right\|_2$ can be expanded and simplified with Jensen's inequality:

$$\begin{aligned}E(q) &= E_\epsilon\left(\sum_{k_1=1}^n\sum_{k_2=1}^n\epsilon_{k_1}\epsilon_{k_2}\left(\mathbf{q}^s\circ\prod_{r=s+1}^L q_1^r\mathbf{\Phi}_{k_1}^s\right)\cdot\left(\mathbf{q}^s\circ\prod_{r=s+1}^L q_1^r\mathbf{\Phi}_{k_2}^s\right)\right)^{\frac{1}{2}}\\ &\le \left(\sum_{k_1=1}^n\sum_{k_2=1}^n E_{\epsilon_{k_1}\epsilon_{k_2}}\epsilon_{k_1}\epsilon_{k_2}\left(\mathbf{q}^s\circ\prod_{r=s+1}^L q_1^r\mathbf{\Phi}_{k_1}^s\right)\cdot\left(\mathbf{q}^s\circ\prod_{r=s+1}^L q_1^r\mathbf{\Phi}_{k_2}^s\right)\right)^{\frac{1}{2}}\\ &= \left\|\sum_{k=1}^n\mathbf{q}^s\circ\prod_{r=s+1}^L q_1^r\mathbf{\Phi}_k^s\right\|_2.\end{aligned}$$

Thus, $K(q)$ can be simplified as follows:

$$\hat{K}(q) \le DB_s\frac{P^{L-s}}{n}\prod_{m=s+1}^{L}C_m\left\|\sum_{k=1}^n\mathbf{q}^s\circ\prod_{r=s+1}^L q_1^r\mathbf{\Phi}_k^s\right\|_2.$$

Based on the prerequisite that only h_s has a dropout operation, we have that $q_1^r=1$ holds for $s+1\le r\le L$. Therefore, we have

$$\hat{K}(q) \le DB_s\frac{P^{L-s}}{n}\prod_{m=s+1}^{L}C_m\left\|\sum_{k=1}^n\mathbf{q}^s\circ\mathbf{\Phi}_k^s\right\|_2.$$

Finally, we estimate the Dropout Rademachder Complexity $K(q)$ based on (10):

$$K(q) \le DB_s\frac{P^{L-s}}{n}\prod_{m=s+1}^{L}C_mE_{\mathbf{x},\mathbf{q}}\left[\sqrt{\left\|\sum_{k=1}^n\mathbf{q}^s\circ\mathbf{\Phi}_k^s\right\|_2^2}\right].$$

Based on Jensen's inequality, we complete our proof and derive the constraint function $U(q)$:

$$K(q) \le DB_s\frac{P^{L-s}}{n}\prod_{m=s+1}^{L}C_mE_{\mathbf{x}}\left[\sqrt{E_{\mathbf{q}}\left\|\sum_{k=1}^n\mathbf{q}^s\circ\mathbf{\Phi}_k^s\right\|_2^2}\right] = U(q).$$

□

3.2. Generalization gap optimization

Based on Theorem 1, the generalization gap can be bounded by the constraint function $U(q)$ based on (14):

$$G = R_{\mathrm{exp}} - R_{\mathrm{emp}} \le 2U(q) + B\sqrt{\ln(2/\delta)/n}, \tag{17}$$

where G refers to the generalization gap and $U(q)$ is given in Theorem 1:

$$U(q) = DB_s\frac{P^{L-s}}{n}\prod_{m=s+1}^{L}C_mE_{\mathbf{x}}\left[\sqrt{E_{\mathbf{q}}\left\|\sum_{k=1}^n\mathbf{q}^s\circ\mathbf{\Phi}_k^s\right\|_2^2}\right]. \tag{18}$$

In (17) and (18), we give an upper-bound estimation of the generalization gap. The term $B\sqrt{\ln(2/\delta)/n}$ can be considered a constant and the generalization gap is indeed constrained by $U(q)$. Therefore, we concentrate on minimizing $U(q)$. In $U(q)$, when the distribution of input $\mathbf{X}$ is fixed, the monotonicity of $E_{\mathbf{x}}[L]$ is the same as the monotonicity of L. Together with the conditions that B_s, P, n and C_m are constants, we conclude that minimizing $U(q)$ is equivalent to minimizing the term $\hat{U}(q)$, where

$$\hat{U}(q) = E_{\mathbf{q}}\left\|\sum_{k=1}^n\mathbf{q}^s\circ\mathbf{\Phi}_k^s\right\|_2^2. \tag{19}$$

Note that $\mathbf{q}^s$ and $\mathbf{\Phi}_k^s$ are both vectors, with the element-wise product operation $\circ$. Moreover, we use q_j^s and $\Phi_k^s(j)$ to represent the j-th element of the two vectors, respectively. Now, $\hat{U}(q)$ is expanded as follows:

$$\hat{U}(q) = E_{\mathbf{q}}\left\|\mathbf{q}^s\circ\sum_{k=1}^n\mathbf{\Phi}_k^s\right\|_2^2 = E_{\mathbf{q}}\left[\sum_{j=1}^{N_s}\left(\left(q_j^s\right)^2 * \sum_{k=1}^n\Phi_k^s(j)^2\right)\right].$$

Then, according to the property of the scaled-Bernoulli distribution, we have

$$E_{\mathbf{q}}[(q_j^s)^2] = (1-p_j^s)*\frac{1}{(1-p_j^s)^2} + p_j^s*0 = \frac{1}{1-p_j^s}.$$

We let $S(j)=\sqrt{\sum_{k=1}^{n}\Phi_k^s(j))^2}$ in the following derivation for convenience. Therefore, the expression of $\hat{U}(q)$ can be simplified by a simple algebra:

$$\hat{U}(q)=\sum_{j=1}^{N_s}\left(E_{\mathbf{q}}[(q_j^s)^2]*S(j)^2\right)=\sum_{j=1}^{N_s}\frac{S(j)^2}{1-p_j^s}.$$

Based on the variant of the Cauchy-Buniakowsky-Schwarz inequality, we further optimize the expression as follows:

$$\hat{U}(q)\geq\frac{\left(\sum_{j=1}^{N_s}S(j)\right)^2}{N_s-\sum_{j=1}^{N_s}p_j^s}. \tag{20}$$

The inequality (20) holds iff $\mathbf{p}^s$ satisfies that

$$\frac{S(1)}{1-p_1^s}=\cdots=\frac{S(N_s)}{1-p_{N_s}^s}=\frac{\sum_{j=1}^{N_s}S(j)}{N_s-\sum_{j=1}^{N_s}p_j^s}. \tag{21}$$

To reach the minimum of $\hat{U}(q)$ given in (20), we let the condition of the Cauchy-Buniakowsky-Schwarz inequality in (21) hold. Therefore, the equality in (20) holds as follows:

$$\min\hat{U}(q)=\frac{\left(\sum_{j=1}^{N_s}S(j)\right)^2}{N_s-\sum_{j=1}^{N_s}p_j^s}. \tag{22}$$

Moreover, in (22), the monotonicity of the minimum of $\hat{U}(q)$ is the same as $\sum_{j=1}^{N_s}p_j^s$. Thus, minimizing $\hat{U}(q)$ can be distilled to minimizing $\sum_{j=1}^{N_s}p_j^s$. Based on (21), we have (23) by simple algebra:

$$p_i^s=1-\frac{\left(N_s-\sum_{j=1}^{N_s}p_j^s\right)}{\sum_{j=1}^{N_s}S(j)}S(i)\quad 1\leq i\leq N_s. \tag{23}$$

Then, considering the condition that $0\leq p_i^s\leq 1$, we obtain the minimum of $\sum_{j=1}^{N_s}p_j^s$:

$$\min\sum_{j=1}^{N_s}p_j^s=N_s-\frac{\sum_{j=1}^{N_s}S(j)}{\max_j S(j)}. \tag{24}$$

Since the minimum of $\sum_{j=1}^{N_s}p_j^s$ is achieved, we can obtain the minimum of $\hat{U}(q)$ together with (24) and (22):

$$\min\hat{U}(q)=\max_j S(j)*\sum_{j=1}^{N_s}S(j). \tag{25}$$

Finally, according to the correspondence between the monotonicity of $\hat{U}(q)$ and $U(q)$, we achieve the optimization of the generalization gap bound:

$$R_{\text{exp}}-R_{\text{emp}}\leq 2U(q)+B\sqrt{\ln(2/\delta)/n}, \tag{26}$$

where $U(q)$ is already minimized by the adaptive choice of dropout rates $\mathbf{q}$:

$$U(q)=E_{\mathbf{x}}\left[DB_s\frac{P^{L-s}}{n}\prod_{i=s+1}^{L}C_i\sqrt{\max_j S(j)*\sum_{j=1}^{N_s}S(j)}\right]. \tag{27}$$

3.3. Rademacher dropout: adaptive dropout algorithm

Based on the previous derivation, an optimized bound of the generalization gap is achieved. This means that we optimize the generalization gap bound by assigning the appropriate values for dropout rates $\mathbf{p}$. Note that the achievement of the optimized risk bound requires that the dropout rates $\mathbf{p}$ should satisfy the equalities in (21) and (24). Therefore, the dropout rates are obtained as follows:

$$p_j^s=1-\frac{\sqrt{\sum_{k=1}^{n}\Phi_k^s(j)^2}}{\max_j\sqrt{\sum_{k=1}^{n}\Phi_k^s(j)^2}}\quad 1\leq j\leq N_s. \tag{28}$$

We give a closed-form solution in (28), which represents a distribution estimation of dropout rates $\mathbf{p}$. Based on the closed-form solution in (28), we propose an adaptive dropout algorithm called Rademacher Dropout (RadDropout) that is outlined in Algorithm 1.

Algorithm 1: Rademacher Dropout.

```
Input: The batch size $d$, and the batch input of layer $h_s$:
    $\mathbf{\Phi}^s=\{\mathbf{\Phi}_1^s,\ldots,\mathbf{\Phi}_d^s\}$, where $\mathbf{\Phi}_k^s=(\Phi_k^s(1),\ldots,\Phi_k^s(N_s))$.
Output: $\hat{\mathbf{\Phi}}^s$.
for $1\leq j\leq N_s$ do
    $S(j)\leftarrow\sqrt{\sum_{k=1}^{d}\Phi_k^s(j)^2}$.
end
for $1\leq j\leq N_s$ do
    $p_j^s\leftarrow S(j)\ /\ \max_j S(j)$.
    $q_j^s\leftarrow scaled-Bernoulli(p_j^s)$.
end
for $1\leq k\leq d$ do
    $\hat{\mathbf{\Phi}}_k^s\leftarrow\mathbf{\Phi}_k^s\circ\mathbf{q}^s$.
end
return $\hat{\mathbf{\Phi}}^s=\{\hat{\mathbf{\Phi}}_1^s,\ldots,\hat{\mathbf{\Phi}}_d^s\}$.
```

In Algorithm 1, we use batch inputs as an approximation to calculate $\mathbf{p}$ for the convenience of implementation. In other words, we calculate $\sqrt{\sum_{k=1}^{d}\Phi_k^s(j)^2}$ instead of $\sqrt{\sum_{k=1}^{n}\Phi_k^s(j)^2}$. Therefore, *RadDropout* generates dropout rates $\mathbf{p}$ of layer h_s only based on the batch inputs $\mathbf{\Phi}^s$, and this calculation can be completed through the process of feed-forward propagation. Moreover, no extra hyper-parameters are introduced by the proposed *RadDropout* and no extra tuning work is required. Overall, after the estimation and optimization of the generalization gap, we finally obtain the *RadDropout* algorithm, which is designed as an adaptive and concise mechanism for the update of dropout rates.

3.4. Rademacher dropout: complexity analysis

The *RadDropout* algorithm calculates dropout rates for layer h_s using the batch inputs $\mathbf{\Phi}^s$. Therefore, the entire complexity of the *RadDropout* algorithm is $\mathcal{O}(2N_sd+2N_s)$. Note that the dropout rates are calculated in linear time with respect to N_s, which indicates that *RadDropout* is indeed a lightweight algorithm.

4. Experiment

In this section, we first perform experiments to make a primary validation on the effectiveness of the proposed algorithm, namely *RadDropout*, on three benchmark datasets in the image recognition area: MNIST,[1] NORB,[2] and SVHN.[3] Then, we further verify the proposed *RadDropout* on two comparatively large and complex

[1] http://yann.lecun.com/exdb/mnist/
[2] https://cs.nyu.edu/~ylclab/data/norb-v1.0-small/
[3] http://ufldl.stanford.edu/housenumbers/

datasets: Cifar-100 dataset[4] and TinyImagenet dataset.[5] Since the two datasets contain more abundant class labels and object variations, their experimental results provide a more persuasive verification of the proposed method. All five datasets are commonly adopted in most existing research, such as [20,23,29,36,37]. Meanwhile, we build a fully connected network for MNIST. For NORB and SVHN, we choose simple network structures modified from Lenet-5.[6] For Cifar-100 and TinyImagenet, we adopt the more complex and widely applied VGG-16 [38] network. In the experiments, we compare the performance of *Rademacher Dropout* (*RadDropout*) with several traditional and state-of-art approaches:

- *Concrete Dropout* (*ConDropout*) [29], which performs the minimization of the KL-divergence based on the deep Bayesian Model.
- *Variational Dropout* (*VarDropout*) [30], which performs the evaluation of the posterior distribution via a novel parametric model.
- *Evolutional Dropout* (*EvoDropout*) [20], which performs the risk bound analysis of the SGD algorithm.
- *Traditional Dropout* (*TraDropout*) [37], which sets the same and fixed dropout rates within one layer as the original dropout strategy.

In the training phase, we report the trend of training loss and the training time of every approach to compare the convergence rates on MNIST, NORB, and SVHN in the primary validation part. Furthermore, we report the trend of recognition accuracy on the validation dataset when training the Cifar-100 and TinyImagenet datasets. In the testing phase, we compare the top prediction accuracy of *RadDropout* with other approaches on the testing dataset for MNIST, NORB, SVHN, and Cifar-100. Since the testing labels of the TinyImagenet dataset are not open to the public, we report the evaluation accuracy on its validation dataset as an alternative. Note that we report the peak testing accuracy of every approach [20]. In addition, for SVHN, NORB, Cifar-100 and TinyImagenet, we add *ConDropout* and *VarDropout* after both the convolutional layers and fully connected layers in view of their regularization mechanism. The other three methods, namely *RadDropout, EvoDropout* and *TraDropout*, are added on the fully connected layers. This configuration ensures a fair comparison for each method to achieve its best performance. All of the experiments are implemented on the tensorflow platform.[7]

4.1. MNIST dataset

MNIST is a widely adopted dataset for classification tasks on handwriting digits [39,40]. It is composed of 60,000 images, of which 50,000 images are for training and the other 10,000 images are for testing. Each image is size-normalized and centered in a fixed-size of 28×28. All of the images are classified into 1 of 10 classes. MNIST represents a baseline of the validation for most deep-learning techniques [41–43] and here we first perform experiments on MNIST to make a basic verification of our method.

To achieve the classification task on MNIST, we build a fully connected network with three hidden layers and each layer has 1,024 neurons. The activation function for each fully connected layer is *relu*. The learning rate starts at 0.0001 and the learning algorithm applied in the network is RMSprop-Optimizer. The initialization of weight follows normal distribution with mean 0 and standard deviation 0.01. In the training phase, we let the network train for 40,000 steps with sufficient learning time. Following the original implementations of each approach [20,29,30,37], we add *RadDropout, TraDropout* and *EvoDropout* after the second fully connected layer and apply *VarDropout* and *ConDropout* to each fully connected layer. Detailed structure information is given in Table 1.

Table 1
Structure of deep network for MNIST.

Layer type	Input size	Output size
Fully connected	784	1024
Fully connected	1024	1024
Fully connected	1024	1024
Fully connected	1024	10

Table 2
Training time (s) and classification accuracies (%) on MNIST dataset. Baseline in table means prediction accuracy of DNN trained without dropout.

Approach	Training time	Prediction accuracy
RadDropout	**94.36**	**98.53**
ConDropout	96.38	98.51
VarDropout	327.48	98.42
EvoDropout	120.17	98.40
TraDropout	93.93	98.37
Baseline	86.09	98.21

In the training phase, we report the training loss of each approach in the form of training curves to analyze the convergence rate of the proposed *RadDropout*. For the reason that the trend curves almost overlap between two approaches in the later stages, we display only the first 1400 steps. As illustrated in Fig. 1, the curves show the remarkable improvement of the convergence rate of the proposed method. This is probably because, compared with other approaches, the proposed method has a superior control on the generalization gap with a compact upper-bound estimation. Therefore, the fluctuation of the training loss is decreased and the convergence process is accelerated.

As analyzed previously, *RadDropout* has a lightweight complexity that is in linear time with respect to the number of neurons in the dropout layer. In Table 2, we report the training time of each approach on MNIST as validation. Meanwhile, to test the generalization ability of *RadDropout*, we compare the prediction accuracy of each approach on the MNIST dataset in Table 2 as well.

As reported in Table 2, we observe that *RadDropout* achieves a high prediction accuracy of 98.53% on the MNIST dataset, which nearly equals that of the hand-tuned *Traditional Dropout*. Among other approaches, *Concrete Dropout* also achieves 98.51% as a desired prediction accuracy. In fact, the baseline approach already has a high prediction accuracy of 98% on the MNIST dataset. However, the three-layer fully connected network on the MNIST dataset indeed has limited generalization ability and each increase of 0.1% accuracy on the baseline represents an improvement. Thus Table 2 shows the improvement of the generalization ability of *RadDropout*. Additionally, Table 2 verifies the lightweight complexity of *RadDropout*. As reported in Table 2, *RadDropout* requires the least time to train the MNIST dataset, namely 94.36 s. Other approaches, such as *VarDropout* and *EvoDropout*, consume much more training time.

4.2. NORB dataset

Intended for experiments in three-dimensional object recognition, the small NORB normalized uniform dataset contains 24,300 training examples and 24,300 testing examples [44]. The toy images in NORB can be divided into five categories. All of the images are resized as 96×96. Sampled by two images under six lighting conditions, NORB requires more meticulous feature

[4] https://www.cs.toronto.edu/~kriz/cifar.html
[5] http://tiny-imagenet.herokuapp.com/
[6] http://yann.lecun.com/exdb/lenet/a35.html
[7] https://www.tensorflow.org

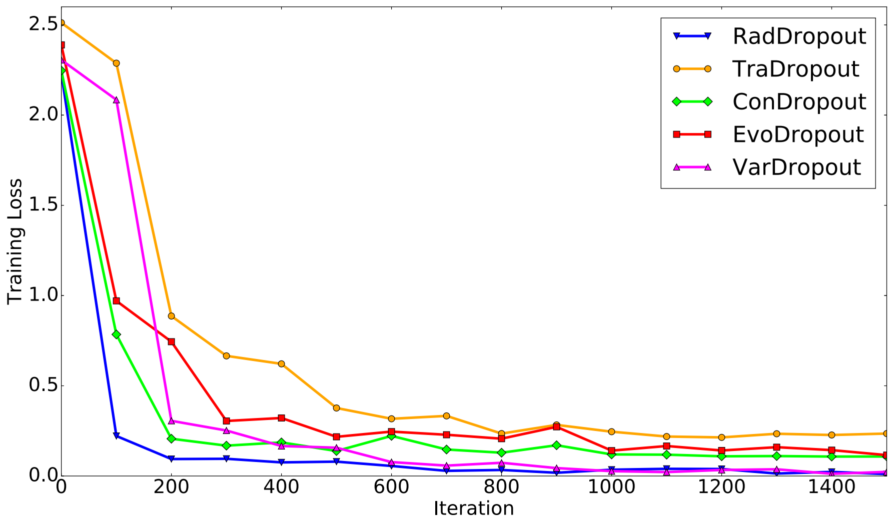


Fig. 1. Training loss on MNIST. This figure illustrates the trend of training loss on MNIST of *RadDropout*, *TraDropout*, *ConDropout*, *EvoDropout* and *VarDropout*. For *ConDropout*, we set the value of the weight-regularizer as 1e-6 and that of the dropout-regularizer as 1e-5. Here *RadDropout* converges to 0.2 as the training loss at the 100th step, while *ConDropout* and *VarDropout* require 200 and 500 steps, respectively, to converge to the same loss value.

learning. Therefore NORB is a more challenging classification task than MNIST, and be used to test the robustness of our approach.

For the NORB dataset, we design a modified six-layer deep network based on the structure of the Lenet-5 network. The network is composed of two convolution layers, two pooling layers and two fully connected layers. The learning rate is fixed at 0.0001 and the learning algorithm applied in the network is Adam-Optimizer. The initialization of weights follows normal distribution with mean 0 and standard deviation 0.1. We train the NORB dataset for 5000 steps for each approach. Among all of the approaches, *ConDropout* and *VarDropout* are applied into each convolutional and fully connected layer [29,30], whereas the other dropout approaches are only added to the first fully connected layer of the network. Detailed structure information is given in Table 3.

Since our network converges quickly on the NORB dataset, here we display only the first 500 steps for sharper contrast. The training curves are presented in Fig. 2. As shown in Fig. 2, *RadDropout* achieves superior performance on convergence rate compared with other approaches. This is probably due to the adaptive choice of dropout rates in *RadDropout* achieving a sharper constraint of the generalization gap. Thus, the training loss of *RadDropout* can avoid getting stuck in a fixed local optimal.

Table 3
Structure of deep network for NORB.

Layer Type	Input	Padding	Kernel Size/Num	Output
Convolution	96*96*2	SAME	5*5/32	96*96*32
Pooling	96*96*32	SAME(2*2)		46*46*32
Convolution	46*46*32	SAME	3*3/64	44*44*64
Pooling	44*44*64	SAME(2*2)		22*22*64
Fully Connected	30,976			1024
Fully Connected	1024			5

To test the generalization ability of *RadDropout* on NORB, we test and compare the prediction accuracy of each dropout approach. As shown in Table 4, *RadDropout* achieves the superior prediction accuracy on NORB, namely 88.77%. The improved generalization ability on NORB of *RadDropout* verifies the robustness of

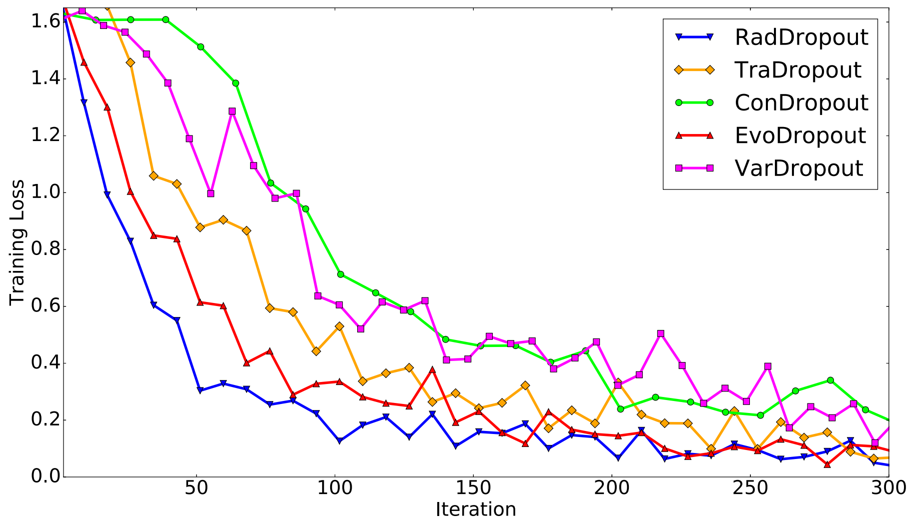


Fig. 2. Training loss on NORB. This figure illustrates the trend of training loss on NORB of *RadDropout*, *TraDropout*, *ConDropout*, *EvoDropout* and *VarDropout*. For *ConDropout*, we set the value of the weight-regularizer as 1e-6 and that of dropout-regularizer as 1e-5. *RadDropout* achieves 0.15 as the training loss at around the 100th step. As the second best approach on NORB, *EvoDropout* requires 170 steps to arrive at 0.15 as the training loss.

Table 4
Training time (s) and classification accuracies (%) on NORB. Baseline means prediction accuracy of DNN trained without dropout.

Approach	Training time	Prediction accuracy
RadDropout	**177.54**	**88.77**
ConDropout	308.46	84.99
VarDropout	683.89	85.63
EvoDropout	181.91	84.80
TraDropout	175.94	84.32
Baseline	170.47	81.03

our proposed *RadDropout*. The improvement of prediction accuracy is probably due to *RadDropout* constraining the generalization error with a sharper upper-bound estimation of the generalization gap. Additionally, the training time reported in Table 4 justifies that *RadDropout* requires less training time than other dropout methods, except for *Traditional Dropout*. Here, *Concrete Dropout* requires much more training time because it requires the collaboration of the regularization of each layer.

4.3. SVHN dataset

As a real-world image dataset for developing machine learning and object recognition algorithms, SVHN can be considered an improved version of the MNIST dataset [45]. It contains 73,257 digit images for training and 26,032 digit images for testing. All of the digit images can be classified into 1 of 10 classes. For the reason that SVHN is obtained as a subset of digits from Google StreetView images, it is also a more challenging task than MNIST. Therefore, we adopt the SVHN dataset to further validate the universality of the proposed *RadDropout*.

The digit images of SVHN have three channels and require more advanced feature extraction. Thus, for the SVHN dataset, we design a more complicated network as a 14-layer network with 9 convolution layers, 3 pooling layers and 2 fully connected layers. The learning rate starts at 0.0001 and learning algorithm is AdamOptimizer. The initialization of weight follows normal distribution with mean 0 and standard deviation 0.1. In the training phase, we let SVHN learn for 18,000 steps. *RadDropout, TraDropout* and *EvoDropout* are added after the first fully connected layer for the SVHN dataset. *VarDropout* and *ConDropout* are applied to each convolutional and fully connected layer. The detailed structure of the network is shown in Table 5.

Table 5
Structure of deep network for SVHN.

Layer type	Input	Padding	Kernel Size/Num	Output
Convolution	32*32*3	SAME	3*3/32	32*32*32
Convolution	32*32*32	SAME	3*3/32	32*32*32
Convolution	32*32*32	SAME	3*3/32	32*32*32
Pooling	32*32*32	SAME(2*2)		16*16*32
Convolution	16*16*32	SAME	3*3/64	16*16*64
Convolution	16*16*64	SAME	3*3/64	16*16*64
Convolution	16*16*64	SAME	3*3/64	16*16*64
Pooling	16*16*64	SAME(2*2)		8*8*64
Convolution	8*8*64	SAME	3*3/128	8*8*128
Convolution	8*8*128	SAME	3*3/128	8*8*128
Convolution	8*8*128	SAME	3*3/128	8*8*128
Pooling	8*8*128	SAME(2*2)		4*4*128
Fully Connected	2,048			512
Fully Connected	512			128
Fully Connected	128			10

Table 6
Training time (s) and classification accuracies (%) on SVHN. Baseline in table means prediction accuracy of DNN trained without dropout.

Approach	Training time	Prediction accuracy
RadDropout	**353.37**	**93.12**
ConDropout	356.81	91.71
VarDropout	520.34	90.85
EvoDropout	361.63	89.79
TraDropout	352.87	90.36
Baseline	349.17	89.40

Here, we display only the first 3000 steps in the training phase to facilitate a comparison of the convergence rates. As shown in Fig. 3, *RadDropout* decreases the training loss at a much faster speed than other approaches as well. Moreover, to verify the algorithm complexity and generalization ability of *RadDropout*, the training time and prediction accuracy for each approach on the SVHN dataset are reported in Table 6. As shown in Table 6, *RadDropout* achieves the desired prediction accuracy compared with other approaches. Meanwhile, with the least training time shown in Table 6, we also further verify the efficiency of *RadDropout*. Therefore, the results on the SVHN dataset further justify the

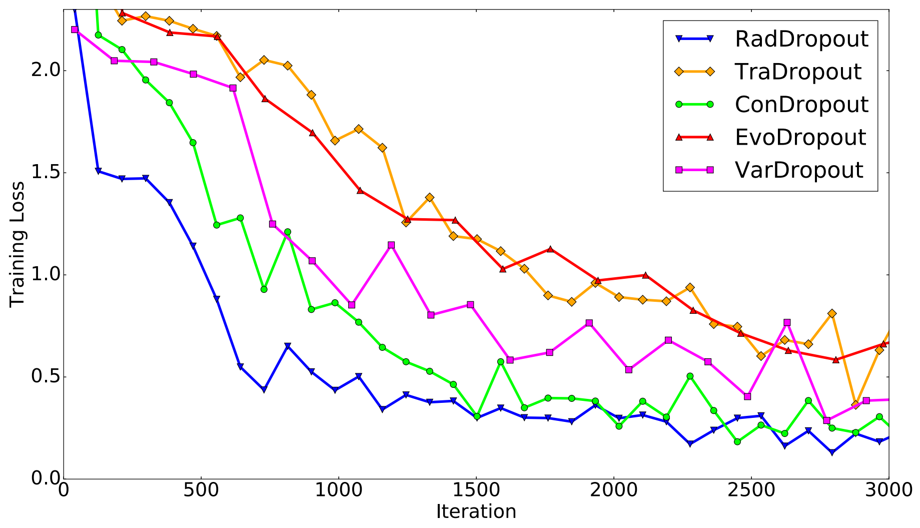


Fig. 3. Training loss on SVHN. This figure illustrates the trend of training loss on SVHN of *RadDropout, TraDropout, ConDropout, EvoDropout* and *VarDropout*. For *ConDropout*, we set the value of the weight-regularizer as 1e-6 and that of the dropout-regularizer as 1e-5. Blue curve implies that *RadDropout* achieves a faster convergence rate than other approaches on SVHN.

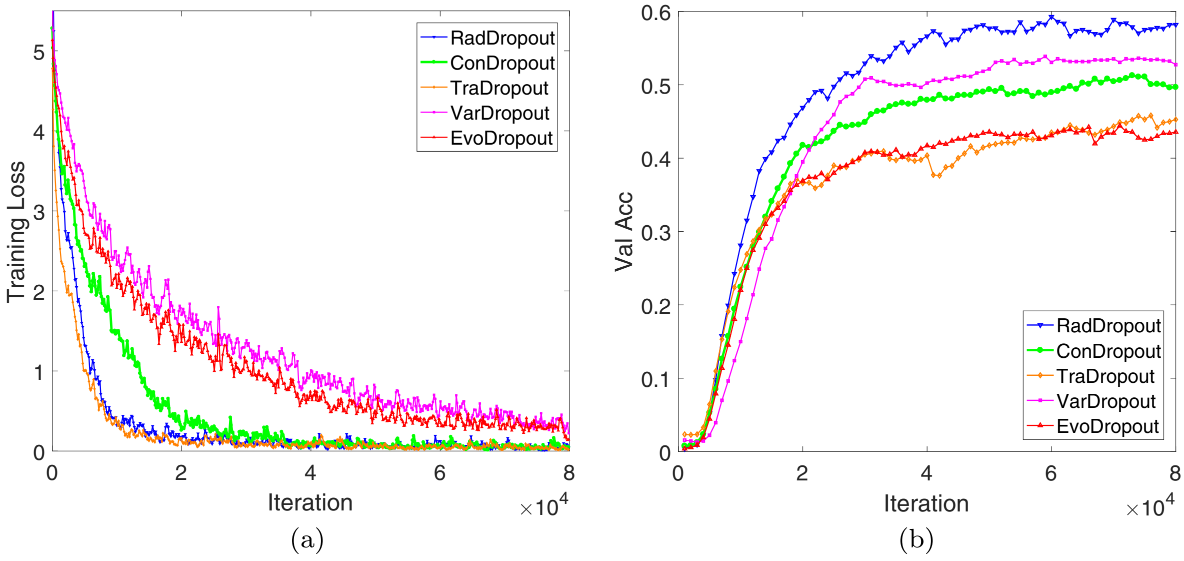


Fig. 4. Training results on Cifar-100 dataset with *RadDropout, ConDropout, TraDropout, VarDropout*, and *EvoDropout*. (a) Training loss and (b) validation accuracy on Cifar-100 dataset. Here, we perform a smoothing operation on the loss curves and accuracy curves to facilitate comparison.

effectiveness of *RadDropout* both on convergence rate and prediction accuracy.

4.4. Cifar-100 dataset

To test the effectiveness of the proposed *RadDropout* on large-scale dataset with complex deep network structure, we further perform experiments on the Cifar-100 dataset. The Cifar-100 dataset consists of 60,000 32×32 color images in 100 classes, with 500 training images and 100 testing images per class. Since the Cifar-100 dataset contains more abundant classes of objects, it is a challenging task to perform recognition tasks on it. Considering the difficulty of the Cifar-100 dataset, we adopt the VGG-16 network with 13 convolution layers, five pooling layers and three fully connected layers. The detailed structure of VGG-16 net is available in [38] and here we set the number of neurons of three fully connected layers to 4096, 4096, and 100, respectively. For the Cifar-100 dataset, *RadDropout, TraDropout* and *EvoDropout* are added after the three fully connected layers. Meanwhile, *VarDropout* and *ConDropout* are applied to each convolutional and fully connected layer. For Cifar-100, the learning rate starts at 0.0001 and the learning algorithm is Adam-Optimizer. Here, we adopt the Xavier initialization method for each layer [33].

In the training phase, we report the training loss to test the convergence speed of *RadDropout*. Moreover, we randomly choose batches from the testing dataset to comprise the validation dataset and report the trend of evaluation accuracy when training Cifar-100. The total training step is 100,000 and we only report the first 80,000 steps to facilitate a comparison. As shown in Fig. 4(a), the proposed *RadDropout* reaches the convergence point at a much faster speed compared to other methods. Moreover, we observe from Fig. 4(b) that the validation accuracy of *RadDropout* also increases at a much faster speed, which indicates that *RadDropout* improves the generalization ability of VGG-16 in an efficient approach.

In the testing phase, we report the final testing accuracy of each method on the Cifar-100 dataset with VGG-16 net. As shown in Table 7, *RadDropout* achieves superior testing accuracy compared to other state-of-the-art dropout methods. Specifically, *RadDropout* achieves a higher accuracy than classical TraDropout, e.g., over 8%. This is mainly attributed to the optimization of the generalization gap by *RadDropout*.

Table 7
Testing accuracies (%) on Cifar-100 dataset with *RadDropout, ConDropout, VarDropout, EvoDropout* and *TraDropout*. Baseline means prediction accuracy of DNN trained without dropout.

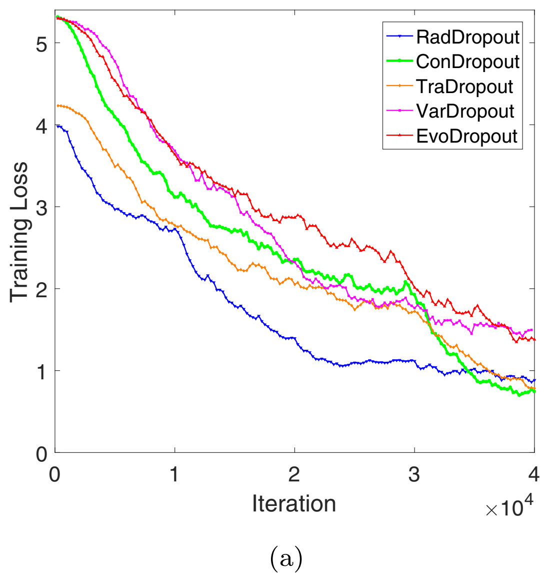
Approach	Testing accuracy
RadDropout	**56.78**
ConDropout	52.82
VarDropout	54.21
EvoDropout	50.58
TraDropout	48.97
Baseline	47.81

4.5. TinyImagenet

As a representative subset of Imagenet,[8] TinyImagenet is comprised of 120,000 64×64 images belonging to 200 categories, with each category having 500 training images, 50 validation images and 50 test images. The categories are synsets of the WordNet hierarchy [46], and the images are similar in spirit to the ImageNet images used in the ILSVRC benchmark [47], but with lower resolution [48]. Since recognition on TinyImagenet is far more difficult than in the other four datasets, we finally test the performance of *RadDropout* on it. For TinyImagenet, we also adopt VGG-16 net with five dropout approaches and make a fair comparison without any other regularization methods. Here, we change the number of neurons of three fully connected layers to 4096, 2048, and 200 in VGG-16. In a similar setting to Cifar-100, we add *RadDropout, TraDropout* and *EvoDropout* after the three fully connected layers, with *VarDropout* and *ConDropout* applied to each convolutional and fully connected layer.

We first report the training loss and validation accuracies of each dropout approach on the TinyImagenet dataset in Fig. 5. As shown in Fig. 5(a) and (b), *RadDropout* has achieved much faster convergence speed than other methods. In the testing phase, we evaluate each method on the validation dataset as an alternative, as the testing labels are not open to the public. As shown in Table 8, *RadDropout* still maintains an ideal evaluation accuracy and outperforms other methods when facing the large-scale dataset.

[8] http://www.image-net.org/



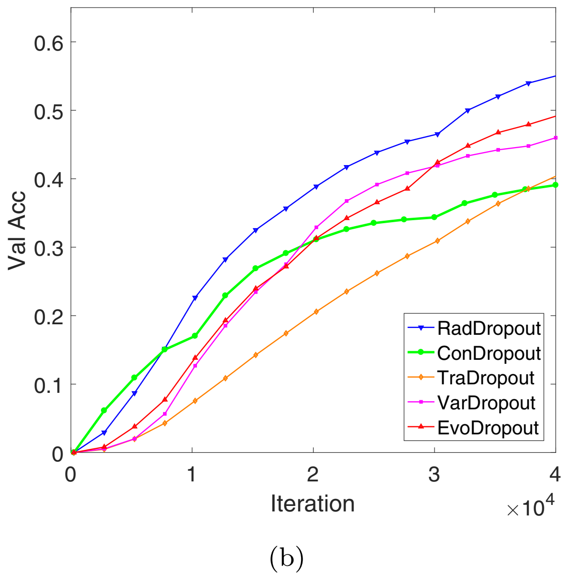


Fig. 5. Training results on TinyImagenet dataset with *RadDropout*, *ConDropout*, *TraDropout*, *VarDropout*, and *EvoDropout*. (a) Training loss and (b) validation accurracy on TinyImagenet dataset. Here, we perform a smoothing operation on the loss curves and accuracy curves to facilitate a comparison.

Table 8
Validation accuracies (%) on TinyImagenet dataset with *RadDropout*, *ConDropout*, *VarDropout*, *EvoDropout* and *TraDropout*. Baseline means prediction accuracy of DNN trained without dropout.

Approach	Validation accuracy
RadDropout	**51.86**
ConDropout	43.52
VarDropout	45.87
EvoDropout	50.21
TraDropout	43.98
Baseline	43.20

5. Conclusions

In this paper, we propose a novel dropout method to achieve the adaptive adjustment of dropout rates. Based on Dropout Rademacher Complexity, we first prove that the generalization gap is bounded by a constraint function related to dropout rates. By a theoretical derivation, we minimize the constraint function and derive a closed-form solution as the distribution estimation of dropout rates. As a result, we propose an adaptive dropout algorithm called Rademacher Dropout based on the closed-form solution with a lightweight complexity. Experimental results on five benchmark datasets verify that the proposed Rademacher Dropout improves both the convergence rate and prediction accuracy. Finally, our study utilizes risk bound analysis to optimize the generalization gap, which provides a significant train of thought for follow-on research. Our planned future work includes the further validation on more challenging dataset (e.g., Imagenet [47]), the derivation of more compact bound estimation of the generalization gap and development of more stable and accurate regularization approaches.

Conflict of Interest

The authors declare that there is no conflict of interests regarding the publication of this article.

Acknowledgement

This work was supported by the National Science Foundation of China (Grant Nos. 91648204 and 11701566).

References

[1] P. Tang, H. Wang, S. Kwong, Deep sequential fusion LSTM network for image description, Neurocomputing 312 (2018) 154–164.
[2] C. Bai, L. Huang, X. Pan, J. Zheng, S. Chen, Optimization of deep convolutional neural network for large scale image retrieval, Neurocomputing 303 (2018) 60–67.
[3] H.-j. Kim, J. Kim, J. Kim, P. Lim, Towards perfect text classification with Wikipedia-based semantic naïve Bayes learning, Neurocomputing (2018).
[4] Y. Cheng, W. Zhang, Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels, Neurocomputing 272 (2018) 63–73.
[5] N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P.T.P. Mudigere, On large–batch training for deep learning: Generalization gap and sharp minima, The International Conference on Learning Representations (ICLR), 2017.
[6] E. Hoffer, I. Hubara, D. Soudry, Train longer, generalize better: closing the generalization gap in large batch training of neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 1731–1741.
[7] K. Kawaguchi, L.P. Kaelbling, Y. Bengio, Generalization in deep learning, arXiv:1710.05468 (2017).
[8] Z. Qi, B. Wang, Y. Tian, P. Zhang, When ensemble learning meets deep learning: a new deep support vector machine for classification, Knowl.-Based Syst. 107 (2016) 54–60.
[9] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, International Conference on Machine Learning, 2015, pp. 448–456.
[10] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv:1207.0580 (2012).
[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[12] B.E. Rosen, Ensemble learning using decorrelated neural networks, Connect. Sci. 8 (3–4) (1996) 373–384.
[13] V. Pham, T. Bluche, C. Kermorvant, J. Louradour, Dropout improves recurrent neural networks for handwriting recognition, in: Proceedings of the 2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2014, pp. 285–290.
[14] S. Wager, W. Fithian, S. Wang, P.S. Liang, Altitude training: Strong bounds for single-layer dropout, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 100–108.
[15] Q. Qian, J. Hu, R. Jin, J. Pei, S. Zhu, Distance metric learning using dropout: a structured regularization approach, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 323–332.

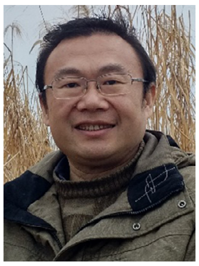
[16] D. Warde-Farley, I.J. Goodfellow, A. Courville, Y. Bengio, An empirical analysis of dropout in piecewise linear networks, The International Conference on Learning Representations (ICLR), 2017.
[17] W. Mou, Y. Zhou, J. Gao, L. Wang, Dropout training, data-dependent regularization, and generalization bounds, in: Proceedings of the International Conference on Machine Learning, 2018, pp. 3642–3650.
[18] R. Shwartz-Ziv, N. Tishby, Opening the black box of deep neural networks via information, arXiv:1703.00810 (2017).
[19] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximagao2016dropouttion, arXiv:1506.02157 (2015).
[20] Z. Li, B. Gong, T. Yang, Improved dropout for shallow and deep learning, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 2523–2531.
[21] W. Gao, Z.-H. Zhou, Dropout rademacher complexity of deep neural networks, Sci. China Inf. Sci. 59 (7) (2016) 072104.
[22] K. Zhai, H. Wang, Adaptive dropout with rademacher complexity regularization (2018).
[23] S. Wager, S. Wang, P.S. Liang, Dropout training as adaptive regularization, in: Proceedings of the Advances in Neural Information Processing Systems, 2013, pp. 351–359.
[24] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 1050–1059.
[25] V. Koltchinskii, D. Panchenko, Rademacher processes and bounding the risk of function learning, in: High Dimensional Probability II, Springer, 2000, pp. 443–457.
[26] X. Wu, J. Zhang, Researches on rademacher complexities in statistical learning theory: a survey, Acta Automat Sinica 43 (1) (2017) 20–39.
[27] J. Ba, B. Frey, Adaptive dropout for training deep neural networks, in: Advances in Neural Information Processing Systems, 2013, pp. 3084–3092.
[28] L. Bottou, Stochastic gradient descent tricks, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 421–436.
[29] Y. Gal, J. Hron, A. Kendall, Concrete dropout, in: Advances in Neural Information Processing Systems, 2017, pp. 3584–3593.
[30] D.P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, in: Advances in Neural Information Processing Systems, 2015, pp. 2575–2583.
[31] Y. Gal, Uncertainty in Deep Learning, University of Cambridge (2016).
[32] C. Gulcehre, M. Moczulski, M. Denil, Y. Bengio, Noisy activation functions, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 3059–3068.
[33] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[34] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.
[35] P.L. Bartlett, S. Mendelson, Rademacher and gaussian complexities: risk bounds and structural results, J. Mach. Learn. Res. 3 (Nov) (2002) 463–482.
[36] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, International Conference on Machine Learning, 2013, pp. 1319–1327.
[37] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.
[38] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, The International Conference on Learning Representations (ICLR), 2014.
[39] L. Deng, The mnist database of handwritten digit images for machine learning research [best of the web], IEEE Signal Process. Mag. 29 (6) (2012) 141–142.
[40] Y. LeCun, The mnist database of handwritten digits, http://yann.lecun.com/exdb/mnist/ (1998).
[41] S. Wang, C. Manning, Fast dropout training, Proceedings of the International Conference on Machine Learning, 2013, pp. 118–126.
[42] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus, Regularization of neural networks using dropconnect, in: Proceedings of the International Conference on Machine Learning, 2013, pp. 1058–1066.
[43] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, The International Conference on Learning Representations (ICLR), 2015.
[44] Y. LeCun, F.J. Huang, L. Bottou, Learning methods for generic object recognition with invariance to pose and lighting, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., 2, IEEE, 2004, pp. II–104.
[45] I.J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, V. Shet, Multi-digit number recognition from street view imagery using deep convolutional neural networks, The International Conference on Learning Representations (ICLR), 2014.
[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009., Ieee, 2009, pp. 248–255.
[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[48] L. Yao, J. Miller, Tiny imagenet classification with convolutional neural networks, CS 231N (2015).

Haotian Wang was born in Anhui, China, in 1995. He received the B.E. degree in computer science and technology from National University of Defense Technology, Changsha, China, in 2017. He is currently pursuing the M.E. degree in computer science and technology with the National University of Defense Technology, Changsha, China. His research interests include machine learning and computer vision.

Yang Wenjing was born in Changsha, China. She received the Ph.D. degree in Multi-scale modelling from Manchester University. She is currently an associate research fellow in State Key Laboratory of High Performance Computing, National University of Defense Technology. Her research interests include machine learning, robotics software and high-performance computing.

Zhenyu Zhao received the B.S. degree in mathematics from University of Science and Technology of China in 2009, and the Ph.D. degree in applied mathematics from National University of Defense and Technology in 2016. Currently, he is a lecturer at College of Liberal Arts and Sciences, National University of Defense and Technology. His research interests include computer vision, pattern recognition and machine learning.

Tingjin Luo received the Ph.D. degree in College of Science from National University of Defense Technology, Changsha, China, in 2018. He has received the B.S. and master's degrees from the College of Information System and Management, National University of Defense Technology, Changsha, China, in 2011 and 2013, respectively. He was a visiting Ph.D. student with the University of Michigan, Ann Arbor, MI, USA, from 2015 to 2017. He is currently an Assistant Professor with the College of Science, National University of Defense Technology. He has authored several papers in journals and conferences, such as IEEE TKDE, IEEE TCYB, IEEE TIP, Scientific Reports and KDD 2017. His current research interests include machine learning, multimedia analysis, optimization, and computer vision.



Ji Wang is a professor in the State Key Laboratory of High Performance Computing, School of Computer, National University of Defense Technology. He received his PhD in Computer Science from National University of Defense Technology. His research interests include programming methodology, formal methods and software engineering for smart systems.

Yuhua Tang received the BS and MS degrees from the Department of Computer Science, National University of Defense Technology, China, in 1983 and 1986, respectively. She is currently a professor in the State Key Laboratory of High Performance Computing, National University of Defense Technology. Her research interests include supercomputer architecture and core router's design.