IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 19, 2019, accepted March 8, 2019, date of publication March 13, 2019, date of current version April 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2904881

β-Dropout: A Unified Dropout

LEI LIU[1], YUHAO LUO[2], XU SHEN[1], MINGZHAI SUN[2], AND BIN LI[1], (Member, IEEE)
[1]CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System, University of Science and Technology of China, Hefei 230026, China
[2]Department of Precision Machinery and Precision Instrumentation, University of Science and Technology of China, Hefei 230026, China

Corresponding author: Bin Li (binli@ustc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61836011 and Grant 61473271, in part by the Fundamental Research Funds for the Central Universities under Grant WK2090090023, and in part by the GPU Computing Cluster of the Data Center, School of Information Science and Technology, University of Science and Technology of China.

ABSTRACT Dropout is an effective regularization method for deep learning tasks. Several variants of dropout based on sampling with different distributions have been proposed individually and have shown good generalization performance on various learning tasks. Among these variants, the canonical Bernoulli dropout is a discrete method, while the uniform dropout and the Gaussian dropout are continuous dropout methods. When facing a new learning task, one must make a decision on which method is more suitable, which is somehow unnatural and inconvenient. In this paper, we attempt to change the selection problem to a parameter tuning problem by proposing a general form of dropout, β-dropout, to unify the discrete dropout with continuous dropout. We show that by adjusting the shape parameter β, the β-dropout can yield the Bernoulli dropout, uniform dropout, and approximate Gaussian dropout. Furthermore, it can obtain continuous regularization strength, which paves the way for self-adaptive dropout regularization. As a first attempt, we propose a self-adaptive β-dropout, in which the parameter β is tuned automatically following a pre-designed strategy. The β-dropout is tested extensively on the MNIST, CIFAR-10, SVHN, NORB, and ILSVRC-12 datasets to investigate its superior performance. The results show that the β-dropout can conduct finer control of its regularization strength, therefore obtaining better performance.

INDEX TERMS Regularization, dropout, deep learning, Gaussian dropout, Bernoulli dropout.

I. INTRODUCTION

Deep neural networks (DNNs) are a popular family of learning models that have shown state-of-the-art performance in many important applications [1]–[6]. However, overfitting, a problem where a model fits well to the training data but lacks generalization for fitting to new data, remains a great challenge. Various regularization techniques have been developed to solve the problem. Dropout method proposed by Bengio *et al.* [7] and Hinton *et al.* [8] is a prevalent and empirically effective method. It works well with nearly any model that can be trained with stochastic gradient descent [7], such as feedforward neural networks, restricted Boltzmann machines [9], and recurrent neural networks [10], [11].

In the canonical binary dropout (Bernoulli dropout), during training, some neuron activities are randomly set to 0 at a probability of 50%. During the test phase, predictions are made by the entire network that contains all the neurons. However, recent studies show that in a real biological system, the response of a neuron is continuous rather than discrete.

Shen *et al.* [12] proposed two continuous dropout methods: the uniform dropout and the Gaussian dropout. As demonstrated in the study [12], the optimal choice of a dropout method is different for different tasks. For example, for the classification task on the MNIST [13] and the CIFAR-10 [14] datasets, the Gaussian dropout achieved the best performance, while for the SVHN [15] dataset, the uniform dropout achieved better performance than Gaussian dropout. This suggests that different tasks require different dropout regularization to achieve the optimal generalization property. In other words, the optimal choice of a dropout method is problem-dependent, as suggested by the no free lunch theorem (NFL) [16]. The next logical question is how to choose an optimal dropout method, and whether there exists a more general dropout method that can adapt to different DNN models.

In this paper, a more general dropout based on the β-distribution is proposed. It unifies the discrete dropout

The associate editor coordinating the review of this manuscript and approving it for publication was Jiankang Zhang.

with continuous dropouts. We show that the β-dropout can yield the Bernoulli dropout and uniform dropout and approximate the Gaussian dropout. In addition, by tuning the shape parameter β, we can control the regularization strength of the dropout to adapt to different problems.

We compare the proposed β-dropout with different commonly used dropout methods on five public datasets, and demonstrate its superior performance on different problems by tuning the regularization strength through the shape parameter β.

The rest of this paper is organized as follows. In Section 2, we provide a brief overview of various dropout methods. In Section 3, we present a detailed description of our proposed β-dropout. In Section 4, we experimentally investigate the performance of the method on MNIST [13], CIFAR-10 [14], SVHN [15], NORB [17] and ImageNet ILSVRC-2012 [18] datasets, and compare them with those of Bernoulli dropout [8], DropConnect [19], Adaptive dropout [20], Gaussian dropout [12], and uniform dropout [12]. Finally, we end the paper in Section 5 with a short conclusion.

II. RELATED WORKS

The Bernoulli dropout was first introduced by Hilton *et al.* [8] and Sivrastava *et al.* [9] to weaken the interdependence between feature detectors in DNNs to mitigate the overfitting problem. In these studies, this method has been confirmed to be effective on different datasets with different deep learning models, and many studies have been conducted on this topic.

Wan *et al.* [19] proposed DropConnect, which suppressed some network connections rather than directly shutting off neural units. This method led to a stronger generalization capability but with much slower training speed. To improve speed, Wang and Manning [21] proposed Fast Dropout, which used the approximate sampling from a Gaussian distribution to eliminate hidden units, and achieved much faster convergence without degrading the accuracy.

In the original dropout, a constant dropout probability was applied to every hidden unit. Ba and Frey [20] proposed an adaptive dropout method, which allowed the possibility of giving different probabilities for different hidden units by overlaying a dropout binary belief network on top of a neural network. The authors showed improved network performance as evaluated using the MNIST and NORB datasets.

Interestingly, instead of computing the dropout rate for each hidden unit, Rennie *et al.* [22] simply changing the dropout rate from a high initial value to zero over the course of training and developed the annealing dropout method, which substantially improved the quality of the model. Later, Kingma *et al.* [23] proposed a variational dropout method. They simply maximized the variational lower bound of the derived dropout's variational objective to learn dropout rates. It was demonstrated that this method often led to better models. Recently, Morerio *et al.* [24] proposed a curriculum dropout. The authors dynamically increased the expected number of suppression hidden units and significantly improved the network generalization capability.

Instead of a Bernoulli distribution, Li *et al.* [25] proposed to use multinomial sampling for dropout, i.e., sampling features or neurons according to a multinomial distribution with different probabilities for different features/neurons. They developed an evolutionary dropout in which the sampling probabilities were computed on-the-fly from a mini-batch. They achieved faster convergence and improved prediction performance. In the work by Li *et al.* [26], the authors proposed a novel adaptive Gaussian noise injection technique (whiteout) to regularize learning of deep neural networks and achieved a much more superior performance than the Bernoulli dropout.

Srinivas and Babu [27] proposed a Generalized dropout method, in which dropout was treated as performing a Bayesian inference over the network parameters. The Generalized dropout places a β prior to adjusting the individual dropout rate for each neuron. In addition, the posterior distribution is modeled by a delta function. Different from [27], the work presented here unifies Bernoulli dropout with continuous dropouts and presents a more general form of dropout (β-dropout). With β-dropout, the problem of selecting the proper dropout method (Bernoulli dropout, uniform dropout or Gaussian dropout, etc.) for a specific learning task is transformed to be a parameter tuning problem within the β-dropout framework, which paves a way to future problem-adaptive dropout regularization.

In this study, we propose a β-dropout that can not only unify the discrete dropout with the continuous dropouts, but also provide tunable regularization strength by adjusting the shape parameter. Through both the theoretical and extensive experimental experiments, we demonstrate the superiority of the proposed β-dropout.

III. THE UNIFIED FRAMEWORK

A. BERNOULLI, UNIFORM AND GAUSSIAN DISTRIBUTIONS CAN BE APPROXIMATED BY A β-DISTRIBUTION

Both continuous dropout and binary dropout can be interpreted within a more general framework of multiplicative noise injection. For a network, the input matrix of a layer is $A^{M\times I}$, the dropout mask matrix (also called injected noise) is $\Xi^{M\times I}$, the output matrix is $Y^{M\times O}$, and the weight matrix is $W^{I\times O}$. Then $Y = AW$. Here we index the elements of these matrices as Y_{mj}, a_{mi} and w_{ij} respectively. Then with the injected noise, the output of the layer is:

$$Y = (A \odot \Xi)W, \text{ with } \xi_{mi} \sim p(\xi) \tag{1}$$

where $\odot$ denotes the element-wise (Hadamard) product and ξ_{mi} is the element of the mask matrix Ξ. Bernoulli dropout can be represented with $\xi_{mi} \sim \text{Bernoulli}(1-p)$, while Gaussian and uniform dropouts are represented by $\xi_{mi} \sim \mathcal{N}(0.5, \sigma^2)$ and $\xi_{mi} \sim U(0, 1)$ respectively.

It is clear that the distribution of the injected noise plays a critical role in dropout. We are looking for a noise distribution with a tunable parameter to regulate the strength of the regularization. Fig. 1 illustrates the dropout process.

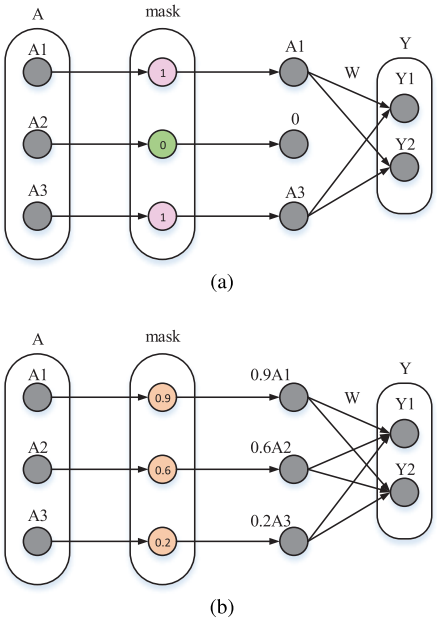


FIGURE 1. Illustration of the dropout process. (a) Bernoulli dropout with the mask sampled from a Bernoulli distribution. (b) Continuous dropout with the mask sampled from a continuous distribution.

We first review the density function of a β-distribution.

$$Beta(x;\alpha,\beta) = \frac{1}{B(\alpha,\beta)}x^{\alpha-1} \times (1-x)^{\beta-1}, \quad (0<x<1, \alpha>0, \beta>0) \tag{2}$$

where α and β are the shape parameters and $B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$. In this study, we set α equal to β so that the distribution is symmetric and has a mean of 0.5. Then $Beta(\alpha,\beta)$ can be abbreviated to $Beta(\beta)$. For Gaussian distribution, the mean is set to 0.5 to have a fair comparison with the Bernoulli dropout [8]. Depending on the shape parameter β, β-distribution can yield Bernoulli, uniform distributions as its special cases and Gaussian-like distribution.

1) When $\beta \to 0$, the β-distribution degenerates to a 2-point Bernoulli distribution [28].

2) When $0 < \beta < 1$, the β-distribution has a U-shape and its variance changes from 1/4 to 1/12.

3) When $\beta = 1$, the β-distribution becomes a 0-1 uniform distribution with a variance of 1/12 [29].

4) When $\beta > 1$, the β-distribution is a symmetric Gaussian-like distribution with a mean of 0.5 and a variance from 0 to 1/12 depending on the value of β. In contrast to the Gaussian distribution, the sampling range of the random variable in the β-distribution is [0,1].

Fig. 2 shows the β-distributions with different shape parameters and the comparisons with Gaussian, Bernoulli and uniform distributions. By adjusting the value of β, the β-distribution can approximate a Bernoulli, a uniform or a Gaussian distribution.

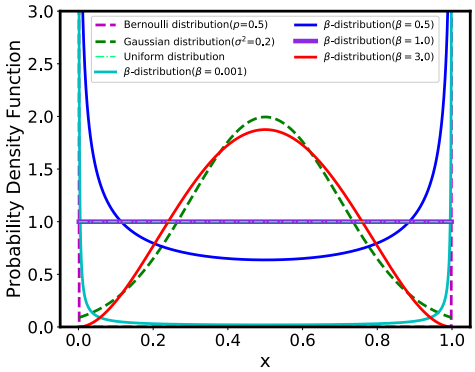


FIGURE 2. The β-distribution with different values of the shape parameter β, together with a Gaussian and a Bernoulli distribution. The β-distribution degenerates to a Bernoulli distribution when β approaches to zero, and it becomes a 0-1 uniform distribution when β is set to 1, and has a Gaussian-like when β is bigger than 1.

B. REGULARIZATION OF THE β-DROPOUT

In this section, we study the static and dynamic properties of the β-dropout and compare it with other continuous dropout methods. Static properties reflect how the dropout affects the output of the network with a fixed set of weights, while the dynamic properties reflect how the dropout changes the learning process of the network [30].

Since Bernoulli dropout achieves the best performance with p set to 0.5 in most situations [8], [12], [31], to match the same expected output of 0.5, we apply $u \sim U(0,1)$ and $g \sim \mathcal{N}(0.5, \sigma^2)$ for uniform and Gaussian dropouts, respectively.

For ease of reading, we derive the static and dynamic properties of the β-dropout using the same symbols as those used in the reference [30]. We skip the duplicate derivations and emphasize only the final results.

1) STATIC PROPERTIES OF THE β-DROPOUT

We first check the static properties of the β-dropout for a single layer of a linear unit and then extend it to nonlinear units.

a: β-DROPOUT FOR A SINGLE LAYER OF LINEAR UNITS

In a single fully connected (FC) *linear* layer, the input $I = [I_1, I_2, \ldots\ldots, I_n]^T$, the weight matrix $W = [\omega_{ij}]_{k\times n}$, and the output $S = [S_1, S_2, \ldots\ldots, S_n]^T$, where the i-th output $S_i = \sum_{j=1}^{n}\omega_{ij}I_j$. The output of the Bernoulli dropout, uniform dropout, Gaussian dropout and β-dropout are denoted as S_i^B, S_i^U, S_i^G, and S_i^β respectively.

$$S_i^B = \sum_{j=1}^{n}\omega_{ij}I_j p_j, \quad E(S_i^B) = \frac{1}{2}\sum_{j=1}^{n}\omega_{ij}I_j \tag{3}$$

$$S_i^U = \sum_{j=1}^{n}\omega_{ij}I_j u_j, \quad E(S_i^U) = \frac{1}{2}\sum_{j=1}^{n}\omega_{ij}I_j \tag{4}$$

$$S_i^G = \sum_{j=1}^{n}\omega_{ij}I_j g_j, \quad E(S_i^G) = \frac{1}{2}\sum_{j=1}^{n}\omega_{ij}I_j \tag{5}$$

$$S_i^\beta = \sum_{j=1}^{n}\omega_{ij}I_j r_j, \quad E(S_i^\beta) = \frac{1}{2}\sum_{j=1}^{n}\omega_{ij}I_j \tag{6}$$

TABLE 1. The variances and covariances of the hidden units of the four different dropout methods. The rate of successful sampling is defined as the probability that the sampled value is in the range of [0,1].

Methods	Parameters	Rate of Successful Sampling (%)	Variance	Covariance
Bernoulli dropout	-	100	0.25	0.25
Uniform dropout	-	100	0.083	0.083
Gaussian dropout	$\sigma^2 = 0.2$	73.64	0.118	0.118
	$\sigma^2 = 0.3$	63.86	0.138	0.138
β-dropout	$\beta \to 0$	100	0.25	0.25
	$\beta = 0.5$	100	0.125	0.125
	$\beta = 1$	100	0.083	0.083

where p_j, u_j, g_j and r_j are random variables with $p_j \sim Bernoulli(0.5)$, $u_j \sim U(0,1)$, $g_j \sim \mathcal{N}(0.5, \sigma^2)$ and $r_j \sim Beta(\beta)$, respectively. The four dropout methods render the same expected output.

The co-adaptation of feature detectors (neurons) is generally reflected by the variance and the covariance of the output units. The large variance of the output of the hidden units generally reflects the diversity of the model, while the covariance between the units at the same layer reveals the independence of the sub-networks [12].

The variance and covariance of the output units of the four dropouts are:

$$\begin{aligned} Var(S_i^B) &= \sum_{j=1}^{n} \omega_{ij}^2 I_j^2 p_j q_j \\ &= \frac{1}{4} \sum_{j=1}^{n} \omega_{ij}^2 I_j^2 \qquad (7) \\ Con(S_i^B, S_k^B) &= \sum_{j=1}^{n} \omega_{ij}\omega_{kj} I_j^2 p_j q_j \\ &= \frac{1}{4} \sum_{j=1}^{n} \omega_{ij}\omega_{kj} I_j^2 \qquad (8) \\ Var(S_i^U) &= \sum_{j=1}^{n} \omega_{ij}^2 I_j^2 var(u_j) \\ &= \frac{1}{12} \sum_{j=1}^{n} \omega_{ij}^2 I_j^2 \qquad (9) \\ Con(S_i^U, S_k^U) &= \frac{1}{12} \sum_{j=1}^{n} \omega_{ij}\omega_{kj} I_j^2 \qquad (10) \\ Var(S_i^G) &= \sum_{j=1}^{n} \omega_{ij}^2 I_j^2 Var(g_j) \\ &= \sum_{j=1}^{n} \omega_{ij}^2 I_j^2 \sigma^2 \qquad (11) \\ Con(S_i^G, S_k^G) &= \sum_{j=1}^{n} \omega_{ij}\omega_{kj} I_j^2 \sigma^2 \qquad (12) \\ Var(S_i^\beta) &= \sum_{j=1}^{n} \omega_{ij}^2 I_j^2 Var(r_j) \\ &= \frac{1}{4(2\beta+1)} \sum_{j=1}^{n} \omega_{ij}^2 I_j^2 \qquad (13) \\ Con(S_i^\beta, S_k^\beta) &= \frac{1}{4(2\beta+1)} \sum_{j=1}^{n} \omega_{ij}\omega_{kj} I_j^2 \qquad (14) \end{aligned}$$

Comparing these four dropout methods, the variances and the covariances of the outputs of β-dropout and Gaussian dropout are controlled by β and σ^2, respectively; while those of the Bernoulli dropout and uniform dropout are not tunable. This enables β-dropout and Gaussian dropout to achieve the best trade-off between the independence of the sub-networks and the model diversity to ensure a good generalization property. Shen *et al.* [12] found that Gaussian dropout works the best with different σ^2 for different tasks. Through our own experiments, we find that the β-dropout achieves the optimal performance by adjusting the parameter β to obtain the most appropriate regularization strength.

To further illustrate that β-dropout can balance the independence of the sub-networks and the model diversity, we performed numerical simulation experiments. We constructed a canonical model as in [8] with the configuration of 784-800-800-10 and the rectified linear unit (ReLU), and applied the network on the MNIST dataset for the classification task.

We characterized the neuron state of the subnetwork as a vector, which in this experiment was a 1600-dimensional vector. For the Bernoulli dropout, this vector was composed of 0 or 1. For continuous dropouts, the values of the elements of the vector were decimals in the range of [0, 1] and sampled from the corresponding distribution. For each dropout method, the experiment was repeated 10000 times. For Gaussian dropout, we applied the two optimal values (0.2 and 0.3) of σ^2, as suggested in [12]. For β-dropout, we compared three different values of the shape parameter, 10^{-3}, 0.5, and 1.0. The variance and covariance of the simulated experiments with different parameters are evaluated and shown in Table 1. The rate of successful sampling is defined as the probability that the sampled value is in the range of [0,1].

As shown in Table 1, the Bernoulli dropout had the largest variance of 0.25 which suggested the largest sub-network diversity; the uniform dropout had the smallest covariance of 0.083, suggesting the strongest sub-network independence.

The β-dropout could achieve the highest sub-network diversity or strongest independence by setting β either to 0 or 1, respectively. Furthermore, the β-dropout can balance independence and diversity by setting the value of β to a decimal between 0 and 1.

For Gaussian dropout, as σ^2 increased, the rate of successful sampling reduced rapidly. Therefore, the Gaussian dropout can only achieve good performance in a small range of the parameter σ^2. In addition, we had to clip the random variable g_j to be in the range of [0, 1] to avoid divergence during propagation [12]. Fortunately, our proposed β-dropout

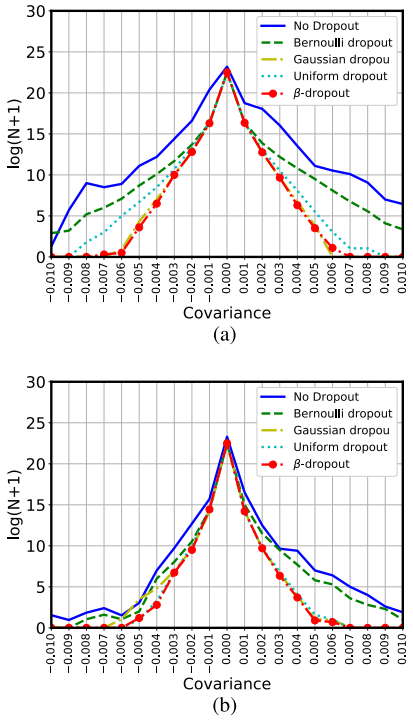


FIGURE 3. Log histogram of covariance between pairs of neurons from the same layer. (a) first layer. (b) Second layer.

overcomes this problem because the range of the random variable r_j is [0, 1].

The covariance of hidden units is related to the co-adaptation of feature detectors. The lower covariance of hidden units indicates that the dropout method has a stronger ability to prevent the co-adaptation of feature detectors. We demonstrated that β-dropout could prevent the co-adaptation of feature detectors by evaluating the distribution of covariance between pairs of neurons in each layer. Fig. 3 shows that among the four dropout methods, the distribution of the covariance of the hidden units in the β-dropout is more concentrated at approximately 0, which indicates that β-dropout achieves better performance than the Bernoulli, Gaussian and uniform dropout in preventing the co-adaptation of feature detectors.

b: β-DROPOUT APPROXIMATION FOR A NONLINEAR UNIT

For a single FC layer, the output of a single nonlinear unit given by the logistic sigmoid function with an input of S is defined as the following:

$$O = sigmoid(S) = \frac{1}{1+ce^{-\lambda S}} \tag{15}$$

For uniform dropout, $S = \sum_{i=1}^{n} \omega_i I_i u_i$, $u_i \sim U(0,1)$. We get $S = \sum_{i=1}^{n} U_i$, $U_i \sim U(0, \omega_i I_i)$, $E[U_i] = \frac{1}{2}\omega_i I_i$ and $Var(U_i) = \frac{1}{12}\omega_i^2 I_i^2$. Obviously, $U_i \leq \max_i \omega_i I_i$, so $S_n^2 = \sum_{i=1}^{n} Var(U_i) \to \infty$. According to the Lyapunov's central limit theorem [32], S approaches to a normal distribution when $n \to \infty$. So, we obtain:

$$\begin{aligned} S &\sim \mathcal{N}(\mu_U, \sigma_U^2), \\ \mu_U &= \sum_{i=1}^{n} E[U_i] = \sum_{i=1}^{n} \tfrac{1}{2}\omega_i I_i, \\ \sigma_U^2 &= \sum_{i=1}^{n} Var(U_i) = \sum_{i=1}^{n} \tfrac{1}{12}\omega_i^2 I_i^2 \end{aligned} \tag{16}$$

For Gaussian dropout, $S = \sum_{i=1}^{n} \omega_i I_i g_i$, $g_i \sim \mathcal{N}(\frac{1}{2}, \sigma^2)$. Similarly, we can infer $S \sim \mathcal{N}(\mu_S, \sigma_S^2)$, where $\mu_S = \frac{1}{2}\sum_{i=1}^{n} \omega_i I_i$, $\sigma_S^2 = \sum_{i=1}^{n} \omega_i^2 I_i^2 \sigma^2$.

For β-dropout, $S = \sum_{i=1}^{n} \omega_i I_i r_i$, $r_i \sim Beta(\beta)$.Similarly, we get $S \sim N(\mu_\beta, \sigma_\beta^2)$, where $\mu_\beta = \frac{1}{2}\sum_{i=1}^{n} \omega_i I_i \mu$, $\sigma_\beta^2 = \sum_{i=1}^{n} \omega_i^2 I_i^2 \, Var(r_i)$, $Var(r_i) = \frac{1}{4(2\beta+1)}$

Since uniform and Gaussian dropouts are special cases of the β-dropout, we only evaluate the expected output of the β-dropout:

$$\begin{aligned} E(O^\beta) &= E[sigmoid(S^\beta)] \\ &= \int_{-\infty}^{\infty} sigmoid(x) N(x|\mu_\beta, \sigma_\beta^2) dx \\ &= sigmoid(\frac{\mu_\beta}{\sqrt{1+\pi Var(r_i)/8}}) \\ &= sigmoid(\frac{\mu_\beta}{\sqrt{1+\pi \frac{1}{4(2\beta+1)}/8}}) \end{aligned} \tag{17}$$

This means that for the β-dropout we have the following recursion:

$$\begin{aligned} E[S_i^h] &= \sum_{l<h} \sum_j w_{ij}^{hl} E[r_j^l] E[O_j^l], \\ E[O_i^h] &\approx sigmoid(\frac{E[S_i^h]}{\sqrt{1+\pi(\sum_{i=1}^{n} \omega_i^2 I_i^2 \frac{1}{4(2\beta+1)})/8}}) \end{aligned} \tag{18}$$

Similarly, for Gaussian dropout, we have the following recursion:

$$\begin{aligned} E[S_i^h] &= \sum_{l<h} \sum_j w_{ij}^{hl} E[g_j^l] E[O_j^l], \\ E[O_i^h] &\approx sigmoid(\frac{E[S_i^h]}{\sqrt{1+\pi(\sum_{i=1}^{n} \omega_i^2 I_i^2 \sigma^2)/8}}) \end{aligned} \tag{19}$$

From [30], we have the following recursion for the Bernoulli dropout:

$$\begin{aligned} E[S_i^h] &= \sum_{l<h} \sum_j w_{ij}^{hl} E[\delta_j^l] E[O_j^l] \\ E[O_i^h] &\approx sigmoid(E[S_i^h]) \end{aligned} \tag{20}$$

By comparing (18), (19) and (20), one can notice the differences of the regularization terms for Gaussian and β-dropouts by the terms of $\sqrt{1+\pi\sum_{i=1}^{n} \omega_i^2 I_i^2 \sigma^2/8}$ and $\sqrt{1+\pi\sum_{i=1}^{n} \omega_i^2 I_i^2 (\frac{1}{4(2\beta+1)})/8}$, respectively. Therefore, both Gaussian and β-dropout can modulate the regularization strength by adjusting the parameters of σ and β.

2) DYNAMIC PROPERTIES OF THE β-DROPOUT

The dynamic properties of the dropout are related to the training procedure and the change of the weights. We choose a simple case of a single linear unit to start and then discuss the nonlinear case. As described in the previous section, the uniform dropout is a special case of the β-dropout. Therefore, we only analyze the dynamic properties of the β-dropout.

a: β-DROPOUT GRADIENT AND REGULARIZATION–SINGLE LINEAR UNIT

In the case of a single linear unit trained with a dropout method, the input is I, the output is $O = S$, and the target is t. Following [30], we use E_{ENS} to denote the ensemble error of all possible sub-networks and E_D to denote the error of the deterministic network with dropout. The typical quadratic of the error is $E = 1/2(t - O)^2$. In the linear case, the ensemble network is identical to the deterministic network obtained by scaling the connections by the dropout probabilities. For a single input I, these error functions are defined by:

$$E_{ENS} = \frac{1}{2}(t - O_{ENS})^2 = \frac{1}{2}(t - \sum_{i=1}^{n} \mu w_i I_i)^2 \tag{21}$$

and

$$E_D = \frac{1}{2}(t - O_D)^2 = \frac{1}{2}(t - \sum_{i=1}^{n} r_i w_i I_i)^2 \tag{22}$$

where μ is the mean value of the β-distribution selector random variables, r_i are the β-distribution selector random variables.

The gradient of the error associated with the ensemble of all possible sub-networks can be evaluated as:

$$\frac{\partial E_{ENS}}{\partial w_i} = \frac{1}{2}(t - O_{ENS})\mu I_i \tag{23}$$

The gradient of error associated with the network with dropout:

$$\begin{aligned}\frac{\partial E_{ENS}}{\partial w_i} &= \frac{\partial E_D}{\partial O}\frac{\partial O}{\partial w_i} = -(t - O)\frac{\partial O}{\partial w_i} = -(t - O_D) r_i I_i \\ &= -t r_i I_i + w_i {r_i}^2 {I_i}^2 + \sum_{i \neq j} w_i r_i r_j I_i I_j \end{aligned} \tag{24}$$

The last vector in (24) is a random vector variable and we can take its expectation. Assuming that the random variables r_is are pairwisely independent, we have

$$\begin{aligned} E[\frac{\partial E_D}{\partial w_i}] &= -t\mu I_i + w_i(\mu^2 + Var(r_i)){I_i}^2 + \sum_{i \neq j} w_i \mu^2 I_i I_j \\ &= \frac{\partial E_{ENS}}{\partial w_i} + w_i {I_i}^2 \frac{1}{4(2\beta + 1)} \end{aligned} \tag{25}$$

Note that the expectation of the gradient with dropout is the gradient of the regularized ensemble error:

$$E_D = E_{ENS} + \frac{1}{2}\sum_{i=1}^{n} {\omega_i}^2 {I_i}^2 \frac{1}{4(2\beta + 1)} \tag{26}$$

In the case of Gaussian dropout, we have:

$$E_D = E_{ENS} + \frac{1}{2}\sum_{i=1}^{n} {\omega_i}^2 {I_i}^2 \sigma^2 \tag{27}$$

From [30], we know that for the Bernoulli dropout we have:

$$E_D = E_{ENS} + \frac{1}{2}\sum_{i=1}^{n} {\omega_i}^2 {I_i}^2 Var(p_i) \tag{28}$$

The regularization term is usually the weight decay based on the square of the weights to ensure that the weights do not become too large [30]. For the β-dropout, the regularization strength ($\frac{1}{2}\sum_{i=1}^{n} {\omega_i}^2 {I_i}^2 \frac{1}{4(2\beta+1)}$) is adjustable so that it can adapt to different model complexity to reduce the overfitting problem. By adjusting the parameter β, the regularization strength varies continuously. When β approaches to $+\infty$, its regularization strength is 0, which corresponds to the method without dropout regularization. Therefore by tuning the value of β, we are able to apply the β-dropout method to different problems.

b: β-DROPOUT GRADIENT AND REGULARIZATION–SINGLE SIGMOIDAL UNIT

For a single sigmoidal unit, we study the dropout gradient and the adaptive regularization properties of different dropout methods. In this section, we skip the duplicate derivations and emphasize only on the final results. For the β-dropout we get:

$$\begin{aligned} E_D &= E_{ENS} + \sum_{i=1}^{n} \lambda \omega_i \mu_i I_i sigmoid'\left(\frac{\mu_\beta}{\sqrt{1 + \pi\, Var(S_i^\beta)/8}}\right) \\ &\quad \times \left(\frac{\frac{\pi}{16}(\sum_j \omega_j \mu I_j)(\omega_i^2 I_i^2 Var(r_i))}{1 + \frac{\pi}{8}\sum_i \omega_i^2 I_i^2 Var(r_i)}\right) \\ &= E_{ENS} + \frac{1}{2}\lambda sigmoid'\left(\frac{\mu_\beta}{\sqrt{1 + \pi\sigma_\beta{}^2/8}}\right) \\ &\quad \times \sum_{i=1}^{n}\sum_j \omega_i \omega_j \mu^2 I_i I_j \left(\frac{\pi}{8}\omega_i^2 I_j^2 \frac{1}{4(2\beta + 1)}\right) \\ &\quad / \left(1 + \frac{\pi}{8}\sum_i \omega_i^2 I_j^2 \frac{1}{4(2\beta + 1)}\right) \end{aligned} \tag{29}$$

For Gaussian dropout:

$$\begin{aligned} E_D &= E_{ENS} + \sum_{i=1}^{n} \lambda \omega_i \mu_i I_i sigmoid'\left(\frac{\mu_S}{\sqrt{1 + \pi\sigma_S^2/8}}\right) \\ &\quad \times \left(\frac{\frac{\pi}{16}(\sum_j \omega_j \mu I_j)(\omega_i^2 I_i^2 \sigma^2)}{1 + \frac{\pi}{8}\sum_i \omega_i^2 I_i^2 \sigma^2}\right) \\ &= E_{ENS} + \frac{1}{2}\lambda sigmoid'\left(\frac{\mu_S}{\sqrt{1 + \pi\sigma_S^2/8}}\right) \\ &\quad \times \sum_{i=1}^{n}\sum_j \omega_i \omega_j \mu^2 I_i I_j \left(\frac{\pi}{8}\omega_i^2 I_j^2 \sigma^2\right) \\ &\quad / \left(1 + \frac{\pi}{8}\sum_i \omega_i^2 I_j^2 \sigma^2\right) \end{aligned} \tag{30}$$

From [30] we know for the Bernoulli dropout:

$$E_D = E_{ENS} + \frac{1}{2}\lambda sigmoid'(U)\sum_{i=1}^{n} {\omega_i}^2 {I_i}^2 Var(p_i) \tag{31}$$

In (29), we can find the β-dropout provide the regularization term $\sum_{i=1}^{n}\sum_j \omega_i \omega_j \mu^2 I_i I_j (\frac{\pi}{8}\omega_i^2 I_j^2 \frac{1}{4(2\beta+1)})/(1 + \frac{\pi}{8}\sum_i \omega_i^2 I_j^2 \frac{1}{4(2\beta+1)}$. Compared with the Bernoulli dropout,

we have the extra regularization terms including covariance between the $\sum_i \omega_i^2 I_i^2 Var(r_i)$, $(I_i I_j)$ and weight $(\omega_i \omega_j)$. On the one hand, the regularization terms can increase interdependent regularization and avoid interdependence and overfitting. On the other hand, by adjusting the shape parameter β, the β-dropout can adjust the strength of regularization (As β increases, the regularization strength gradually decreases). We can always find an optimal regularization strength that is well-suited to the particular task we want to solve.

IV. EXPERIMENTAL INVESTIGATION AND DISCUSSION

In the previous section, we proved that the proposed β-dropout could unify the Bernoulli dropout and the uniform dropout, as well as approximate the Gaussian dropout. In addition, the β-dropout provides continuous regularization strength to adapt to different tasks by selecting an optimal value of the parameter β.

In this section, we first use the MNIST dataset [13] to systematically study the performance of the proposed β-dropout, especially the effects of the shape parameter of the distribution. Then we fully evaluate the performance of the β-dropout with classification tasks on CIFAR-10 [14], SVHN [15], NORB [14] and ImageNet ILSVRC-2012 [18]. We compare its performance with the Bernoulli dropout [8], Adaptive dropout [20], DropConnect [19], Gaussian dropout [12], and uniform dropout [12].

For the Bernoulli dropout [8] and the DropConnect [19], the dropout rate p is set to the most commonly used value of 0.5 [9]. For the Adaptive dropout, the parameters of α and β are selected from the set of {-1, 0, 1} and {-0.5, 0, 0.5}, respectively [33]. For Gaussian dropout, the mean is set to 0.5 and σ^2 is selected from {0.2, 0.3}, as suggested in [12]. We clip Gaussian dropout variable to be in the range of [0, 1] ($g_i = 1$ if $g_i \geq 1$ and $g_i = 0$ if $g_i \leq 0$) to alleviate the divergence during propagation [12]. For uniform dropout, u_i is subject to $U(0, 1)$. For the β-dropout, the shape parameter β is set to values drawn from the set {0.001, 0.1, 0.2, 0.5, 1.0, 3.0}. Following [12], we use THEANO [34] to implement the feedforward neural networks that only consist of fully connected (FC) layers. The convolutional neural networks are implemented in Caffe [35].

To perform the statistical comparison, we repeat each experiment N times and evaluate the mean errors and the standard deviations. For the MNIST, CIFAR-10, SVHN, and NORB datasets, N is set to 30, and for ImageNet ILSVRC-2012, N is set to 10 due to the high computational cost.

A. EXPERIMENTS ON MNIST

The MNIST dataset consists of 70,000 handwritten digit (from 0 to 9) images of size 28×28, among which 60,000 images are used for training while the rest are used for testing. During the training process, 50,000 images from the training set are randomly selected as the training images while the remaining 10,000 are used as validation images. We reproduced the results in [12] and applied the same settings to the β-dropout including a linear momentum schedule, a constant weight constraint, and an exponentially decaying learning rate.

TABLE 2. Performance comparison of various dropout methods on MNIST dataset. No data augmentation was applied in the experiment.

Method	Error(%)	
	Sigmoid	ReLU
No dropout	1.58±0.045	1.15±0.032
Bernoulli dropout	1.35±0.044	1.06±0.037
DropConnect	1.37±0.067	1.01±0.043
Adaptive dropout	1.30±0.022	1.02±0.026
Gaussian dropout	1.15±0.024	0.95±0.030
Uniform dropout	1.21±0.031	0.96±0.038
β-dropout	**1.12**±0.032	**0.94**±0.036

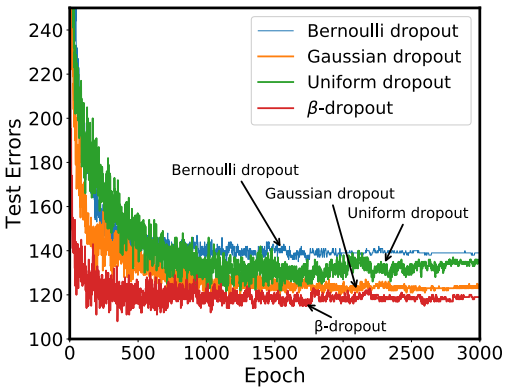


FIGURE 4. Test errors versus the number of epochs for different methods, including experiments without dropout (No dropout), with the Bernoulli dropout, Gaussian dropout, uniform dropout, or the β-dropout on MNIST. No data augmentation was applied in this experiment. (MNIST, 784-800-800-10, Sigmoid).

1) β-DROPOUT OUTPERFORMS OTHER CONTINUOUS DROPOUTS

In the experiment, we construct an MLP network with two fully connected (FC) layers, each consisting of 800 nodes. The final network model is 784-800-800-10, which is the same as the model in [8]. We apply either the sigmoid or ReLU activation function. During training, no data augmentation is applied. From Table 2, we can see that the β-dropout outperformed other dropout techniques irrespective of whether sigmoid or ReLU activation function was applied.

We further test the dependence of the test errors on the number of epochs of the different dropout methods. As shown in Fig. 4, all the continuous dropouts (uniform, Gaussian and the β-dropouts) achieve lower test errors than the discrete Bernoulli dropout. Among the continuous dropouts, the β-dropout outperformed both uniform and Gaussian dropouts, suggesting that the β-dropout has better generalization properties than the other dropout methods.

The number of nodes in the hidden layers is another important factor that affects the performance of a network. We vary the number of nodes in the hidden layer and evaluate the test error using different dropout methods. Fig. 5 shows that

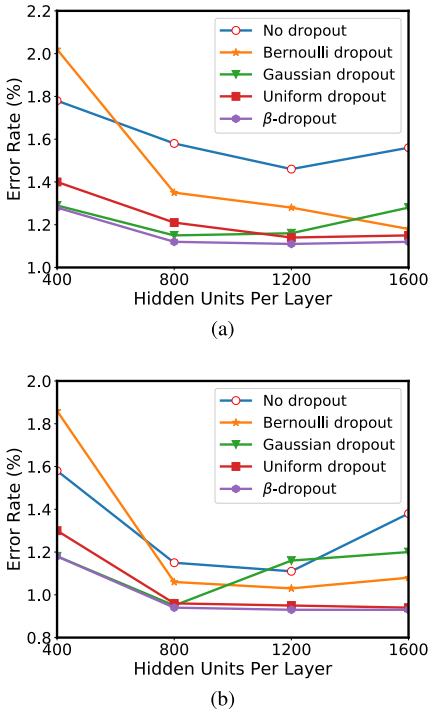


FIGURE 5. Change of test error as a function of the number of nodes in the hidden layers with different dropout methods. (a) A network using Sigmoid activation function with 784-*x*-*x*-10 architecture, where *x* is the number of nodes in the hidden layers. (b) A network using the ReLU activation function with a 784-*x*-*x*-10 architecture.

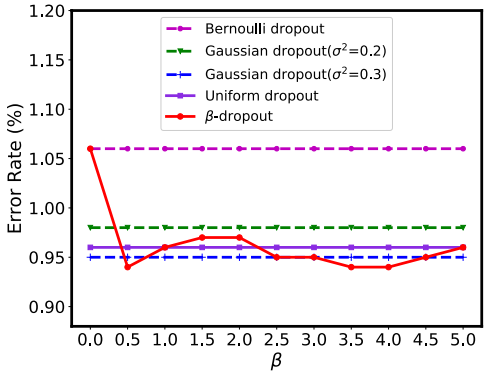


FIGURE 6. Comparisons of the Bernoulli, Gaussian, uniform, and β-dropouts, and the dependence of the error rate on the value of β.

as the model size increases the test error decreases. Our β-dropout performs as well as or better than other dropout methods under all tested conditions, irrespective of the activation functions. The difference is especially noticeable for smaller networks.

2) INFLUENCE OF THE SHAPE PARAMETER β IN β-DROPOUT

From (11) - (14), it can be seen that for the β-dropout, there is an extra degree of freedom to balance network output and model complexity. To investigate how β affects the performance of the networks, we train the network (784-800-800-10) on MNIST with the ReLU activation function. For the β-dropout, β is selected from the set {0.001, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0}. Fig. 6 shows

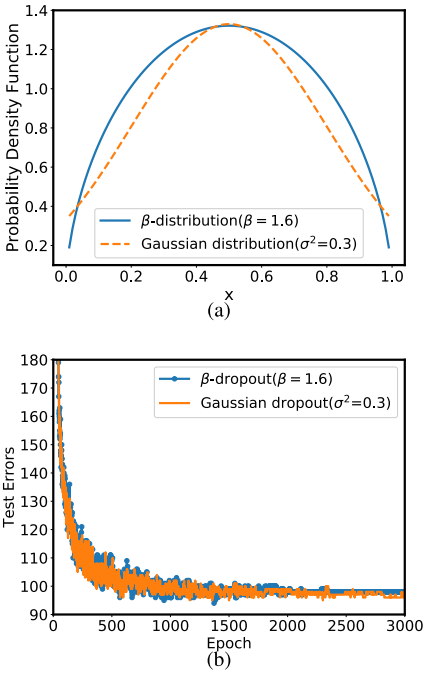


FIGURE 7. The network achieves comparable test errors with Gaussian and the β-dropout methods when the β distribution curve approaches Gaussian curve by adjusting the value of β (MNIST, 784-800-800-10, ReLU). (a) By adjusting the β value, the curve of the β distribution approaches to the curve of Gaussian distribution. (b) The test error curves as evaluated with Gaussian dropout method and β-dropout method overlap on each other.

the comparisons of the Bernoulli dropout, uniform dropout, and Gaussian dropouts with σ^2 set to 0.2 and 0.3, the best variances as suggested in [12]. By selecting the right β value, β-dropout outperforms all other three dropout methods, and achieves the lowest error rate when β is set to 0.5, 3.5 and 4.0. In addition, as β approaches 0, it exhibits the same performance as the Bernoulli dropout, and when β is set to 1, it exhibits the same performance as the uniform dropout. When β is greater than 1, the sampling distribution of our β-dropout is Gaussian-like and exhibits a performance very close to that of the Gaussian dropout.

3) THE BEST PERFORMANCE ACHIEVED BY GAUSSIAN DROPOUT CAN BE ACHIEVED BY β-DROPOUT, BUT NOT VICE VERSA

By adjusting the shape parameter β, the β-distribution can approximate the Bernoulli, Gaussian and uniform distributions. From the previous section, we conclude that Gaussian dropout achieves the best performance with σ^2 set to 0.3. By adjusting β to 1.6, the β-distribution can approximate the Gaussian distribution, as shown in Fig. 7(a). As expected, the β-dropout achieved similar performance as Gaussian dropout (Fig. 7(b)) as evaluated with the testing error. This suggests that the shape of the distribution determines the performance of the dropout methods. By adjusting the β parameter, the shape of the β-distribution can change from a U-shape to a Gaussian-like(Fig. 2). However, Gaussian distributions cannot have this U-shape. From Table 3, we can clearly see

TABLE 3. Correspondence of different dropout methods with different shape parameters β of β-dropout.

β	Distribution Shape	Dropout Methods
$\beta \to 0$	Binary point	Bernoulli dropout
$0 < \beta < 1$	U-shape	U-shaped dropout
$\beta = 1$	Flat	Uniform dropout
$1 < \beta < +\infty$	Gaussian-like	Gaussian-like dropout
$\beta \to +\infty$	Dirac delta	No dropout

the correspondence between the different distribution shapes of β-dropout and different dropout methods. From Fig. 6, it is concluded that when β is set to 0.5, the network achieves the best performance, better than the performance achieved with Gaussian dropout. As shown in Fig. 2, when β is 0.5, the β-distribution curve has a U-shape, which cannot be approximated by a Gaussian distribution. This demonstrates that β-dropout can scan a much larger multifold space than Gaussian dropout. What can be achieved by Gaussian dropout could be achieved by the β-dropout, but not vice versa.

To further illustrate that the proposed β-dropout is more robust than the Gaussian dropout, we check the performance of the networks by choosing the values of σ^2 and β near the optimal values of 0.3 and 1.6, respectively. For the Gaussian dropout, we chose five different values: 0.1, 0.2, 0.3, 0.4, and 0.5. For the β-dropout, we selected 1.4, 1.5, 1.6, 1.7, and 1.8 for the parameter β. We evaluated the mean of the performance and the standard deviation with selected parameters for the Gaussian and β-dropouts. The results are shown in Table 4. Although the average error rate of the β-dropout is the same as that of the Gaussian dropout, the standard deviation is much smaller, which suggests that the performance of the network is less sensitive to the values of the shape parameter β.

4) EFFECT OF VARIOUS EXPECTED VALUES IN β-DROPOUT

In the original work, Hinton *et al.* [8] introduced the dropout layer with a rate set to 0.5, which became commonly used later on. Shen *et al.* [12] configured the uniform and Gaussian dropouts to have the same expected value of 0.5 to match with the Bernoulli dropout. In the previous sections, we applied the same strategy by setting $\alpha = \beta$ in the β-distribution, and the sampling range $U[0, 1]$ in the uniform distribution to avoid introducing more adjustable parameters.

In this section, to explore the effect of different expected values in the dropout methods, the expected value was allowed to vary from 0.5. For the Gaussian distribution, μ and σ^2 can be tuned in $N(\mu, \sigma^2)$. For the uniform distribution, the distribution is extended from $U(0, 1.0)$ to $U(a, b)$ with the expected value (mean) of $\frac{b-a}{2}$. Meanwhile, all the values of $U(a, b)$ are in the range $[0, 1]$ (a reasonable distribution for dropout mask variables), with $0 \leq a < b \leq 1$. For the β-distribution, the expected value of a β-distribution random variable is $\frac{1}{1+\beta/\alpha}$. The β-distribution, depending on the parameters α and β, can also yield Bernoulli and uniform distributions as its special cases and approaching a Gaussian-like distribution with different expected values.

1) When $\alpha \to 0$, $\beta \to 0$, the β-distribution degenerates to a 2-point Bernoulli distribution [28]. By setting the expected value of the β-distribution and the Bernoulli distribution to be equal, ($\frac{1}{1+\beta/\alpha} = p$), we can obtain the condition that β-distribution turns to a Bernoulli distribution with p. Thus, when $\alpha \to 0$, $\beta \to 0$ and $\alpha = (\frac{p}{1-p})\beta$, a β-distribution can be equivalent to a Bernoulli distribution with p.
2) When $\alpha = 1$, $\beta = 1$, the β-distribution turns to be a standard uniform distribution ($U(0, 1)$). When we limit the β-distribution in the range [a,b], the β-distribution turns to $U(a, b)$.
3) When $\alpha > 1$, $\beta > 1$, the β-distribution becomes a Gaussian-like distribution. By adjusting the parameters α and β, the β-distribution can approximate a wide spectrum of Bell-like distributions.

To explore the effect of various mean values, the three types of Dropouts under distributions with means of 0.4, 0.5 and 0.6 on the MNIST dataset were applied for the classification task with the canonical model configured as 784-800-800-10 and ReLU [8]. To ensure that all four dropout algorithms achieve the same expected output of 0.4, 0.5 and 0.6, the parameter settings are shown in Table 5. For the Bernoulli dropout, the p value is set to 0.4, 0.5 and 0.6, respectively while for the Gaussian dropout, the mean is set to 0.4, 0.5 and 0.6, with σ^2 being selected from the best variance of {0.2, 0.3}. In the uniform dropout, μ_i are subject to $U(0, 0.8)$, $U(0, 1.0)$ and $U(0.2, 1.0)$ respectively. For the β-dropout, α are set to $\frac{2}{3}\beta$, β and $\frac{3}{2}\beta$ respectively, and β is set to values ranging from {0.001, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0}. For additional considerations, the β-distribution is clip to [0, 0.8] and [0.2, 1.0] for the mean of 0.4 and 0.6, respectively, when β is set to 1.0.

To offer more intuitionistic information concerning distribution with various expected values, the Gaussian, uniform and β-distributions with expected values of 0.4, 0.5 and 0.6 are drawn, as shown in Fig. 8. Compared across (a), (b) and (c) in Fig. 8, it can be found that the β-distribution approaches the Gaussian and uniform distributions with various expected values by selecting the appropriate α and β or limiting the range of random variables.

Fig. 9 shows that the β-dropout outperforms the Bernoulli, Gaussian and uniform dropouts with three various mean values. In addition, the β-dropout has a similar performance as the Bernoulli dropout as β approaches 0, and similar performance as the uniform dropout when β is set to 1. Interestingly, when the expected value was set to 0.6, the performance of the Bernoulli, Gaussian, uniform and β-dropouts all perform better than when the expected value was set to 0.4 or 0.5, suggesting that the best expected value for dropouts on the network and task is 0.6.

Exploring the optimal expectations of dropouts or best results under a certain task is not the purpose of this paper; rather, we are concerned with the ability to unify the framework of discrete and continuous dropouts. We can conclude that, under various expected values, adjusting the α

TABLE 4. Performance comparison of various dropout methods on MNIST dataset. No data augmentation was applied in the experiment.

	Gaussian Dropout(σ^2)	β-dropout(β)
Values of the parameters	0.1,0.2,0.3,0.4,0.5	1.4,1.5,1.6,1.7,1.8
Mean of the performance (Error Rate)	0.978	0.978
Standard deviation of the performance	0.021	0.003

TABLE 5. The parameter settings of the Bernoulli, Gaussian, uniform and β-dropouts with expected values of 0.4, 0.5 and 0.6.

Distributions \ Means / Methods	0.4	0.5	0.6
Bernoulli dropout	Bernoulli(0.4)	Bernoulli(0.5)	Bernoulli(0.6)
Gaussian dropout	$N(0.4, 0.2)$, $N(0.4, 0.3)$	$N(0.5, 0.2)$, $N(0.5, 0.3)$	$N(0.6, 0.2)$, $N(0.6, 0.3)$
Uniform dropout	$U(0, 0.8)$	$U(0, 1.0)$	$U(0, 1.0)$
β-dropout	$Beta(\frac{2}{3}\beta, \beta)$	$Beta(\beta, \beta)$	$Beta(\frac{3}{2}\beta, \beta)$

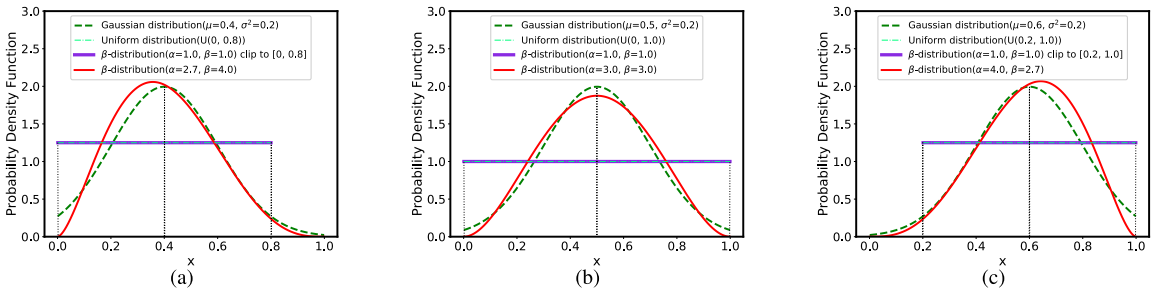


FIGURE 8. The β-distribution with expected values of 0.4 (a), 0.5 (b) and 0.6 (c), together with a Gaussian and a uniform distribution.

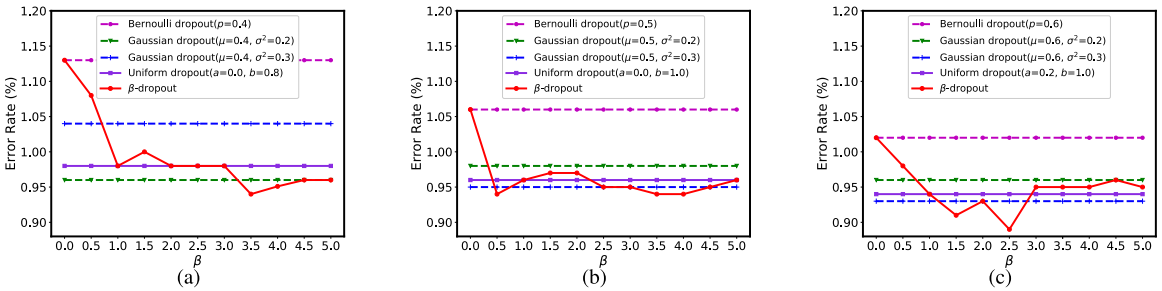


FIGURE 9. The performance curves of the β-dropout with expected values of 0.4 (a), 0.5 (b) and 0.6 (c), compared with the best results of other dropout methods with same expected values. (MNIST, 784-800-800-10, ReLU).

and β parameters in our β-dropout can still yield Bernoulli and uniform dropouts and approximate Gaussian dropout. However, a mean of 0.5 is still a common and valid choice in most cases in regards to a new task. To be fair and simple, the expected value of 0.5 was still used in all subsequent experiments for all dropout methods.

5) β-DROPOUT OUTPERFORMS OTHER CONTINUOUS DROPOUTS IN MORE COMPLEX NETWORKS

To further test the effectiveness of the proposed β-dropout, we apply it to a more complex network as proposed in [19]. Different from the simple MLP structure, this network consists of 2 convolution layers with 32-64 feature maps and kernels of the size of 3×3 in each layer. The second convolution layer is followed by an FC layer with 150 units. ReLU activation function is used in this network. This network is initialized with $\mathcal{N}(0, 0.01)$, and the input is the original pixels without any pre-processing. The learning rate is initialized with 0.01 and was manually decreased by a multiplier of 0.5 or 0.1 when the loss function of the validation error reaches a plateau. As shown in Table 6, the β-dropout achieves statistically significant improvement over all baseline methods.

B. EXPERIMENTS ON CIFAR-10

In the CIFAR-10 dataset, there are 50,000 training and 10,000 testing images of 10-classes of natural 32×32 RGB images. We use a global contrast normalization and a zero-phase component analysis proposed in [36] to preprocess all the image. We apply the modified network-in-network model as in [12]. Different dropout methods are applied only to the first FC layer. During training, the weights are fine tuned

TABLE 6. Performance comparison of various dropout methods on MNIST dataset using CNN network. No data augmentation was applied in the experiment.

Method	Error(%)
No dropout	0.674±0.047
Bernoulli dropout	0.551±0.022
DropConnect	0.581±0.012
Adaptive dropout	0.591±0.017
Gaussian dropout	0.534±0.009
Uniform dropout	0.549±0.023
β-dropout	**0.527**±0.017

from the pre-trained weights in [37] to evaluate different dropout methods. The learning rate is initialized to 0.01 and decayed by 10 every 3,000 iterations. No data augmentation is applied during training. The test results of the model after 10,000 iterations are presented in Table 7. The proposed β-dropout achieves the lowest test error.

TABLE 7. Performance comparison on CIFAR-10.

Method	Error(%)
No dropout	10.65±0.114
Bernoulli dropout	10.55±0.065
DropConnect	10.40±0.178
Adaptive dropout	10.46±0.081
Gaussian dropout	10.18±0.137
Uniform dropout	10.47±0.142
β-dropout	**10.09**±0.108

In addition, we evaluate the performance of the classification network for each class. To this end, we construct the confusion matrix (Fig. 10). Compared with the Bernoulli dropout, the Adaptive Dropout, DropConnect, uniform Dropout, and Gaussian dropout, the β-dropout achieves the best performance on 8, 8, 9, 8, and 8 classes, respectively. The improved performance is due to the better generalization property of the β-dropout.

C. EXPERIMENTS ON SVHN

SVHN [15] consists of 630,420 RGB images of street view house numbers. The size of the images is 32×32, and 26,032 of the images are for testing and the rest are for training. The goal is to classify the digits in the images. The images are preprocessed by local contrast normalization [38] to reduce the variation of the colors and brightness. In addition, the images are cropped to a size of 28×28 pixels and are randomly rotated and scaled as done in [39]. Following [12], we randomly flip images and then introduce 15% scaling and rotation variations.

We apply a model consisting of two convolutional layers and two locally connected layers [40]. An FC layer consisting of 512 neurons with ReLU activation is added in front of the softmax layer.

As shown in Table 8, the β-distribution and uniform distribution achieve the same performance and are better than all other continuous dropout methods. In this experiment, for β-dropout, the dropout mask is sampled from *Beta*(1.0).

TABLE 8. Performance comparison on SVHN.

Method	Error(%)
No dropout	2.09±0.002
Bernoulli dropout	2.00±0.016
DropConnect	1.96±0.005
Adaptive dropout	2.13±0.235
Gaussian dropout	1.93±0.016
Uniform dropout	1.92±0.012
β-dropout	**1.92**±0.012

As described earlier, *Beta*(1.0) is the same as a 0-1 uniform distribution. In other words, when β is 1, the β-dropout degenerated to a uniform dropout.

D. EXPERIMENTS ON NORB

The twofold NORB (Generic Object Recognition in Images, jittered-cluttered) dataset consists of 29,160 images for training and 58,320 images for testing. These images are on different random backgrounds and are categorized into six different classes. We apply the same network as in the SVHN experiment. The images are first downsampled from 108 × 108 to 48 × 48 and augmented by 15% rotation and scaling. The learning rate of the model is initialized to 0.01. Table 9 summarizes the results using different dropout methods. Our proposed β-dropout significantly outperforms all other dropout methods. The improved performance is due to the better choice of the regularization strength in the β-dropout.

TABLE 9. Performance comparison on NORB.

Method	Error(%)
No dropout	3.55±0.070
Bernoulli dropout	3.33±0.136
DropConnect	3.49±0.059
Adaptive dropout	3.49±0.235
Gaussian dropout	3.15±0.116
Uniform dropout	3.29±0.184
β-dropout	**3.13**±0.152

E. EXPERIMENTS ON ILSVRC-2012

The ILSVRC-2012 [18] dataset contains images of 1,000 classes, of which 1.3 M images are for training, 50K images are for validation and 100K are for testing.

We apply the VGG-16 model as proposed in [39]. During training, the weights are initialized with weights obtained in [39]. The input images are resized to 224 × 224 and are zero-centered by subtraction of [103.939, 116.779, 123.68] from the BGR channels. The batch size is set to 64 and the optimizer is SGD with the momentum of 0.9. The weight decay is set to 5×10^{-4} and the gradient is clipped to 35. The learning rate is initially set to 10^{-4}, and then decreases by a factor of 10 after 50,000 iterations. To evaluate the network performance, both the top-1 and top-5 errors are used.

During the test phase, the first FC layer and the last two FC layers are replaced by a 7×7 convolutional layer and a 1×1 convolutional layer, respectively. The output of the network is a class score map, which is then spatially

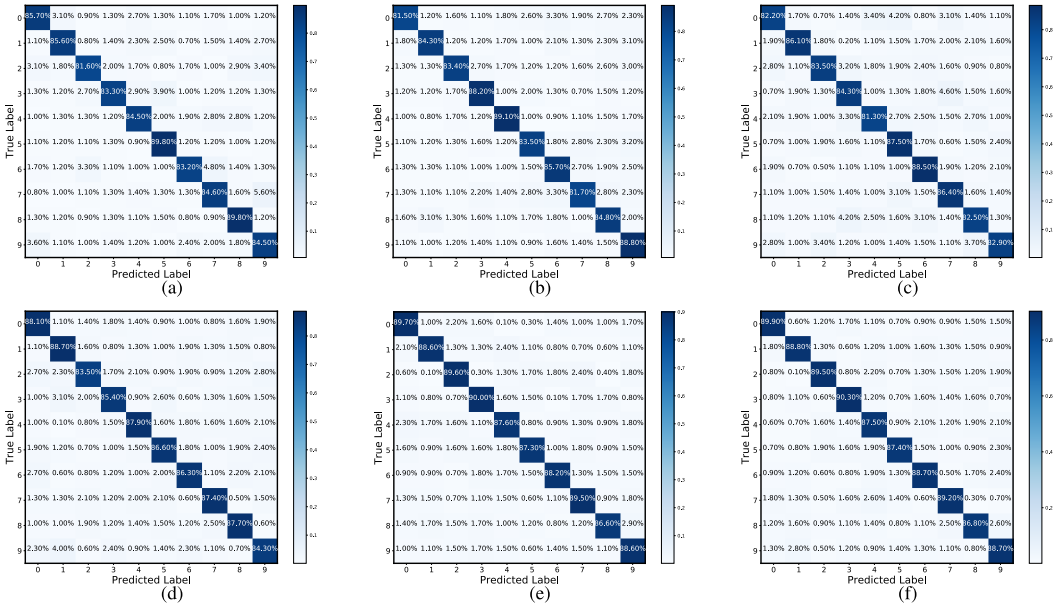


FIGURE 10. Confusion matrices of all six methods. (a) Bernoulli dropout. (b) Adaptive dropout. (c) DropConnect. (d) Uniform dropout. (e) Gaussian dropout. (f) β-dropout. β-dropout achieved the best performance on five classes among all six methods. Specifically, β-dropout achieved higher classification accuracy on 8, 8, 9, 8, and 8 classes than Bernoulli dropout, Adaptive dropout, DropConnect, uniform dropout, and Gaussian dropout, respectively.

TABLE 10. Performance comparison on ImageNet ILSVRC-2012 (mean top-1/top-5 error and standard derivation).

Method	ConvNet Config	Error(%)	
		top-1	top-5
No dropout	VGG_ILSVRC_16_layers	28.36±0.063	9.64±0.012
Bernoulli dropout		26.99±0.065	8.86±0.042
DropConnect		26.82±0.050	8.56±0.037
Adaptive dropout		26.27±0.046	8.41±0.061
Gaussian dropout		25.79±0.045	7.99±0.065
Uniform dropout		25.91±0.046	8.08±0.048
β-dropout		**25.75**±0.052	**7.96**±0.046

TABLE 11. Performance rank of various dropout methods on all five data sets.

Method	MNIST	CIFAR-10	SVHN	NORB	ILSVRC-2012	Average rank
Bernoulli dropout	4	6	5	4	6	5
DropConnect	5	3	4	6	5	4.6
Adaptive dropout	6	4	6	5	4	5
Gaussian dropout	2	2	3	2	2	2.2
Uniform dropout	3	4	1	3	3	2.8
β-dropout	1	1	1	1	1	1

averaged (sum-pooled). We also apply horizontal flipping to augment the image, and the final score is the average of the soft-max results of the original and the flipped images as proposed in [39].

Table 10 shows the results of all the listed dropout algorithms. For a large dataset, β-dropout achieves the best results compared to other algorithms. It is easier to achieve a good model with a large dataset. However, for a complex model, it inevitably leads to similarities between neurons (features). The β-dropout was a simple and effective way to address this problem. It improves the performance of the model by reducing the covariance between neurons, especially when the model is over-fitted in the training data set.

We rank the dropout methods by the performance on each of the five data to summarize the overall performance of various dropout methods. As shown in Table 11, the proposed β-dropout is ranked first.

We note that on the SVHN dataset, β-dropout achieves the same performance as uniform dropout because in this case

TABLE 12. Performance on MNIST using the self-adaptive β-dropout with different t_p and β_0.

β_0	t_p	Error(%)
0.001	500	0.96±0.007
	1000	0.95±0.019
	1500	**0.92**±0.011
	2000	0.97±0.012
	2500	0.98±0.006
0.5	500	0.99±0.007
	1000	0.91±0.019
	1500	**0.88**±0.011
	2000	0.91±0.026
	2500	0.91±0.017

TABLE 13. Performance comparison on CIFAR-10, SVHN, NORB and ImageNet ILSVRC-2012.

Method	Error(%)				
	CIFAR-10	SVHN	NORB	ILSVRC-2012 (top-1)	ILSVRC-2012 (top-5)
Previous optimal methods (exclude β-dropout)	10.18±0.137	1.92±0.012	3.15±0.116	25.79±0.045	7.99±0.065
β-dropout	10.09±0.108	**1.92**±0.012	3.13±0.152	25.75±0.052	7.96±0.046
Self-adaptive β-dropout	**10.07**±0.087	1.93±0.016	**3.10**±0.096	**25.69**±0.027	**7.95**±0.042

β takes the value of 1. In other words, β-dropout is adjusted to a uniform dropout for optimal performance. In contrast, Gaussian dropout cannot achieve the same performance on the SVHN dataset as uniform dropout, as shown in the previous study [12].

V. SELF-ADAPTIVE β-DROPOUT: AN EXAMPLE

Since the β-distribution can be shaped to approximate various distributions by tuning the parameter β, a new line of research can be opened on how to find a way to self-adaptively tune β for a new task that might be quite different from those that have been previously applied. As a first attempt, we propose a self-adaptive β-dropout, in which the parameter β is tuned automatically following a strategy similar to deterministic annealing [22].

According to the result obtained in Section III, we know that β is inversely proportional to the regularization strength (we only consider the case of $\alpha = \beta$ in the β-dropout). With the gradual increase of β, the regularization strength is weakened and the model capacity is increased. By analyzing the evolution process of the β-distribution as β increases, we designed a segmented parameter tuning procedure for β as follows:

$$\beta(t) = \begin{cases} \dfrac{t}{t_p}(1-\beta_0) + \beta_0, & t < t_p \\ \dfrac{t-t_p}{T-t_p}(\beta_{max}-1) + 1, & t \geqslant t_p \end{cases} \tag{32}$$

where β_{max} is the maximum β value and T is the maximum number of epochs for the entire training. t_p is the number of epochs when β reaches 1. In the experiments, β_{max} and T are set to 1000 and 3000 respectively. β_0 is selected from $\{0.001, 0.5\}$ and t_p is selected from $\{500, 1000, 1500, 2000, 2500\}$. We applied the parameter tuning schedule (32) in the β-dropout, and obtained a self-adaptive β-dropout. The new self-adaptive β-dropout is tested on a model with the configuration of 784-800-800-10 and ReLU for MNIST classification task.

Table 12 shows the results of the self-adaptive β-dropout with different t_p. When β_0 is set to 0.001 and t_p is set to 1500, the performance achieves an error of 0.92, which outperforms the best result with an error of 0.94 as reported in Table 2. When β_0 is set to 0.5 and t_p is greater than 1000, the performance all outperforms the best result of 0.94. This suggests that the newly proposed self-adaptive β-dropout is a simple and highly effective way to improve performance.

To further verify the effectiveness of our self-adaptive β-dropout, we applied it to CIFAR-10, SVHN, NORB and ILSVRC-2012 with the same network and hyper-parameter configuration in Section IV. In the experiments, β_0 is set to values from $\{0.001, 0.5\}$, and t_p is selected from $\{500, 1000, 1500, 2000, 2500\}$. The results are reported in Table 13. Here, it can be observed that the proposed self-adaptive β-dropout achieves the best performance on CIFAR-10, NORB and ILSVRC-2012. The experimental results demonstrate the effectiveness of our proposed self-adaptive β-dropout.

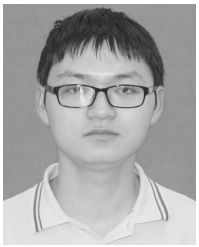
VI. CONCLUSION

Dropout is an effective regularization technique for improving the performance of deep neural networks. The original and most commonly used Bernoulli dropout applies a binary mask to the hidden neurons. However, real neuron activation is random and continuous. Inspired by this phenomenon, continuous dropouts have been developed. Here, we extended the continuous dropout and developed a unified framework – β-dropout. Through theoretical analysis, we prove that the β-dropout could effectively reduce the interdependence between units and the complexity of a model. It can prevent overfitting by increasing the regularization term. Furthermore, compared with other continuous dropout methods, β-dropout can take on additional distribution forms by adjusting the shape parameter β so that the regularization strength can be continuously tuned. Through our experiments, we demonstrate the best performance of a network with a well-adjusted β value. Our β-dropout can unify discrete dropout with continuous dropout. In addition, we propose a self-adaptive β-dropout and demonstrate that it yields better performance than β-dropout in most tasks. In future work, we will further explore a more efficient self-adaptive strategy for tuning parameters.

REFERENCES

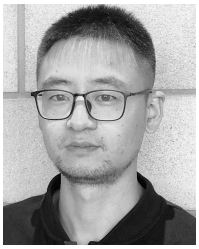
[1] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, Feb. 2018, Art. no. 7068349.
[2] D. Shen, G. Wu, and H. Suk, "Deep learning in medical image analysis," *Annu. Rev. Biomed. Eng.*, vol. 19, pp. 221–248, Jun. 2017.
[3] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5934–5938.
[4] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 4845–4849.
[5] Y. Goldberg, "Neural network methods for natural language processing," *Synth. Lectures Hum. Lang. Technol.*, vol. 10, no. 1, pp. 1–309, 2017.
[6] T. Young, D. Hazarika, S. Poria, and E. Cambria. (2017). "Recent trends in deep learning based natural language processing." [Online]. Available: https://arxiv.org/abs/1708.02709
[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[10] J. Bayer and C. Osendorfer. (2014). "Learning stochastic recurrent networks." [Online]. Available: https://arxiv.org/abs/1411.7610
[11] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. (2013). "How to construct deep recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1312.6026
[12] X. Shen, X. Tian, T. Liu, F. Xu, and D. Tao, "Continuous dropout," *IEEE Trans. neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 3926–3937, 2018.
[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[14] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, vol. 1, no. 4, Tech. Rep., 2009.
[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS workshop Deep Learn. Unsupervised Feature Learn.*, vol. 2, 2011, pp. 1–9.
[16] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[17] Y. Lecun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun./Jul. 2004, pp. II–97–104.
[18] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[19] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066.
[20] J. Ba and B. Frey, "Adaptive dropout for training deep neural networks," in *Proc. Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3084–3092.
[21] S. Wang and C. Manning, "Fast dropout training," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 118–126.
[22] S. J. Rennie, V. Goel, and S. Thomas, "Annealed dropout training of deep networks," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, 2014, pp. 159–164.
[23] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2575–2583.
[24] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino. (2017). "Curriculum dropout." [Online]. Available: https://arxiv.org/abs/1703.06229
[25] Z. Li, B. Gong, and T. Yang, "Improved dropout for shallow and deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2523–2531.
[26] Y. Li, R. Xu, and F. Liu. (2016). "Whiteout: Gaussian adaptive noise regularization in deep neural networks." [Online]. Available: https://arxiv.org/abs/1612.01490
[27] S. Srinivas and R. V. Babu. (2016). "Generalized dropout." [Online]. Available: https://arxiv.org/abs/1611.06791
[28] J. B. S. Haldane, "A note on inverse probability," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 28, no. 1. Cambridge, U.K.: Cambridge Univ. Press, 1932, pp. 55–61.
[29] T. Bayes and R. Price, "An essay towards solving a problem in the doctrine of chances. By the late rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S," *Philosoph. Trans. Roy. Soc. London*, vol. 53, pp. 370–418, Dec. 1763.
[30] P. Baldi and P. Sadowski, "The dropout learning algorithm," *Artif. Intell.*, vol. 210, pp. 78–122, May 2014.
[31] N. Srivastava, "Improving neural networks with dropout," *Univ. Toronto*, vol. 182, p. 566, Jan. 2013.
[32] E. L. Lehmann, *Elements of Large-Sample Theory*. New York, NY, USA: Springer, 2004.
[33] J. Bayer, C. Osendorfer, D. Korhammer, N. Chen, S. Urban, and P. van der Smagt. (2013). "On fast dropout and its applicability to recurrent networks." [Online]. Available: https://arxiv.org/abs/1311.0701
[34] J. Bergstra *et al.*, "Theano: A CPU and GPU math compiler in Python," in *Proc. 9th Python Sci. Conf (SciPy)*, vol. 1, pp. 3–10, Jun. 2010.
[35] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[36] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. (2013). "Maxout networks." [Online]. Available: https://arxiv.org/abs/1302.4389
[37] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400
[38] M. D. Zeiler and R. Fergus. (2013). "Stochastic pooling for regularization of deep convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1301.3557
[39] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[40] A. Krizhevsky. *Cuda-Convnet*. Accessed: 2012. [Online]. Available: https://code.google.com/p/cuda-convnet/



LEI LIU is currently pursuing the Ph.D. degree in information and communication engineering with the School of Information Science and Technology, University of Science and Technology of China (USTC), Hefei, China. His major research interests include deep learning, statistical learning, evolutionary computation, computer vision, and medical image processing their applications in industry.

YUHAO LUO is currently pursuing the Ph.D. degree with the Department of Precision Machinery and Precision Instrumentation, University of Science and Technology of China (USTC), Hefei, China. His major research interests include deep learning, intelligent information processing, computer vision, and medical image processing and their applications in industry.



XU SHEN received the B.S. and Ph.D. degrees from the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application Systems, University of Science and Technology of China, Hefei, China, in 2012 and 2017, respectively. His current research interests include multimedia, computer vision, and deep learning.

MINGZHAI SUN received the Ph.D. degree in physics from the University of Missouri, USA, in 2008. He is currently a Professor with the Department of Precision Machinery and Precision Instrumentation, University of Science and Technology of China (USTC), Hefei, China. His research interests include optical super-resolution microscopy, biomedical device development, and bioimage informatics, particularly on medical image analysis and deep learning-based algorithm development.

BIN LI (M'07) received the B.Sc. degree from the Hefei University of Technology, Hefei, China, in 1992, the M.Sc. degree from the Institute of Plasma Physics, Chinese Academy of Sciences, Hefei, in 1995, and the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, in 2001, where he is currently a Professor with the School of Information Science and Technology. He has authored or co-authored over 40 refereed publications. His current research interests include computation intelligence, optimal design, pattern recognition, and human-computer interaction. He is also a member of the Technical Committee of the Electronic Circuits and Systems Section of CIE and a Senior Member of the Chinese Institute of Electronics (CIE). He is also the Founding Chair of the IEEE Computational Intelligence Society Hefei Chapter and a Counselor of the IEEE USTC Student Branch.