2019 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, OCT. 13–16, 2019, PITTSBURGH, PA, USA

SOFT DROPOUT AND ITS VARIATIONAL BAYES APPROXIMATION

Jiyang Xie[1], *Zhanyu Ma*[1,*], *Guoqiang Zhang*[2], *Jing-Hao Xue*[3], *Zheng-Hua Tan*[4], *Jun Guo*[1]

[1]Pattern Recognition and Intelligent Systems Lab., Beijing University of Posts and Telecommunications, China

[2]School of Electrical and Data Engineering, University of Technology Sydney, Australia

[3]Department of Statistical Science, University College London, United Kingdom

[4]Department of Electronic Systems, Aalborg University, Denmark

ABSTRACT

Soft dropout, a generalization of standard "hard" dropout, is introduced to regularize the parameters in neural networks and prevent overfitting. We replace the "hard" dropout mask following a Bernoulli distribution with the "soft" mask following a beta distribution to drop the hidden nodes in different levels. The soft dropout method can introduce continuous mask coefficients in the interval of $[0, 1]$, rather than only zero and one. Meanwhile, in order to implement the adaptive dropout rate via adaptive distribution parameters, we respectively utilize the half-Gaussian distributed and the half-Laplace distributed variables to approximate the beta distributed masks and apply a variation of variational Bayes optimization called stochastic gradient variational Bayes (SGVB) algorithm to optimize the distribution parameters. In the experiments, compared with the standard soft dropout with fixed dropout rate, the adaptive soft dropout method generally improves the performance. In addition, the proposed soft dropout and its adaptive versions achieve performance improvement compared with the referred methods on both image classification and regression tasks.

Index Terms— Neural networks, soft dropout, beta distribution, Bayesian approximation

1. INTRODUCTION

Recently, neural networks have attracted great attention from academic and industry for their excellent performance on various tasks including image classification [1, 2, 3, 4, 5], image retrieval [6, 7, 8, 9], speech recognition [10, 11], and time series prediction [12, 13] with large-scale datasets. Neural networks including fully connected (FC) neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) are becoming deeper and deeper to extend the capability to learn more discriminative nonlinear patterns in the training process, which may lead to the major drawback of overfitting the limited training data and tend to be difficult in generalization [14].

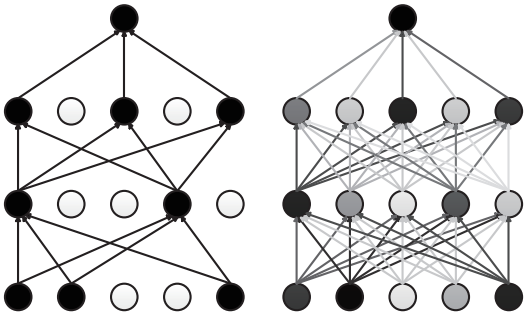
To address this issue, different works focused on the solutions about how to improve the generalization ability and prevent overfitting of deep models, including early stopping [15], data augmentation [16], and regularization [17, 18, 19]. Dropout [17], a fundamental regularization technique, regularizes the model parameters by dropping them randomly in the training steps, which plays an important role in preventing feature co-adaptation [20]. In addition, distinct works [14, 21, 22, 23, 24, 25] created variations of dropout and obtained significant improvement. Wang and Manning [21] proposed a Gaussian approximation called Gaussian dropout with virtually identical regularization performance but much faster convergence than the standard dropout in [18]. Maeda [22] introduced a Bayesian interpretation to optimize the dropout rate which is beneficial for model training and prediction, but focused on the binary variant. Kingma et.al. [14] proposed variational dropout with dropout rate optimized by the stochastic gradient variational Bayes (SGVB) inference [26] leading to much faster convergence than the Gaussian dropout. Gal and Ghahramani [23] predicted the model uncertainty in neural networks via dropout which can be interpreted as a Bayesian approximation in regression, classification, and reinforcement learning. Gal et al. [24] introduced the concrete dropout, an alternative method for automatically tuning the dropout rate. In addition, a dropout variant has been proposed for RNNs focusing on time dependence representation and demonstrated outstanding effectiveness [25]. One thing in common among all the aforementioned dropout techniques is that they all interpret the dropout via Bernoulli or Gaussian prior.

In this paper, integrating advantages of both, we propose a dropout variant named *soft dropout* which can be considered as a generalization of the discrete dropout technique

*Corresponding author.

This work was supported in part by National Key Research and Development Program of China under Grant 2018YFC0807205, the National Natural Science Foundation of China (NSFC) No. 61922015, 61773071, in part by the, in part by the Beijing Nova Program No.Z171100001117049, by the Beijing Nova Program Interdisciplinary Cooperation Project No. Z181100006218137, in part by the Fundamental Research Funds for the Central University No. 2018XKJC02, in part by the scholarship from China Scholarship Council (CSC) under Grant CSC No. 201906470049, and in part by BUPT Excellent Ph.D. Students Foundation No. CX2019109, XTCX201804.

978-1-7281-0824-7/19/$31.00



(a) Standard dropout (b) Soft dropout

Fig. 1: Examples of (a) standard dropout and (b) the proposed soft dropout. Notation: different grey scales represent different values between 0 (by white), *i.e.*, fully "drop" and 1 (by black), *i.e.*, fully "hold".

with Bernoulli dropout mask in [17]. Replacing the masks by the beta distributed variables which are continuously valued in the interval of $[0, 1]$, the proposed soft dropout not only guarantees the fundamental objective of dropout, *i.e.*, ignoring part of model parameters with normalized weights when training, but also samples from a wider space for parameter selection than the discrete dropout since each parameter has more (in principle infinite) states, which can be considered as model ensemble from a wider parameter space. Figure 1 illustrates the difference between the FC neural networks of the discrete dropout and the proposed soft dropout. The soft dropout masks perform various levels (by grey scale in Figure 1) of dropout for the parameters, rather than only 1 (by black) and 0 (by white). We expect the proposed soft dropout can also improve model performance and show reduction of overfitting.

Furthermore, a Bayesian approximation of the proposed soft dropout is introduced for adaptive dropout rate learning by optimizing the prior parameters. It is known that adaptation of the dropout rate affects the performance of the model to some extent [14]. Adaptive dropout rate is beneficial to the optimization of the soft dropout technique. Given the fact that the beta distribution cannot be directly handled by the SGVB algorithm, we utilize the half-Gaussian and the half-Laplace distributions to approximate the beta prior of the soft dropout. The adaptive soft dropout techniques, called Gaussian and Laplace soft dropout respectively, can learn the prior parameters by using the SGVB algorithm and demonstrate better performance than the soft dropout with fixed dropout rate.

2. METHODOLOGY

2.1. Soft Dropout

For a neural network with L layers, we define $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_l\}_{l=1}^{L}$ where $\boldsymbol{\theta}_l \in R^{K_{l-1} \times K_l}$ is the model parameter matrix for the l^{th} layer and $K_l, l = 1, \cdots, L$ is the hidden node number of the l^{th} layer. Note that the inputs are the 0^{th} layer and the outputs are the L^{th} layer. In the standard dropout, a set

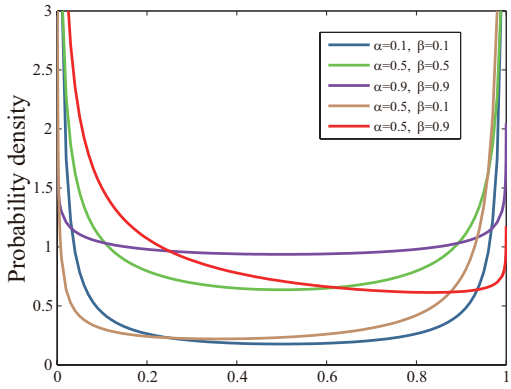


Fig. 2: Probability density functions (PDFs) of beta distribution with different parameters. We only illustrate the cases that $\alpha, \beta < 1$ as examples.

of independent random variables $\boldsymbol{M} = \{\boldsymbol{m}^{(l)}\}_{l=1}^{L}, \boldsymbol{m}^{(l)} = [m_1^{(l)}, \cdots, m_{K_l}^{(l)}]^{\mathrm{T}}$ are sampled from the Bernoulli distribution and considered as the masks multiplied element-wise with the hidden nodes [17].

In the view of Bayesian optimization, the standard dropout can be considered as a regularization with the discrete Bernoulli prior. In this case, the soft dropout is a generalized dropout by replacing the discrete prior with a continuous prior. Specifically, we apply the beta distribution as the prior distribution.

Due to the independent identically distributed elements of the K_l-dimensional mask $\boldsymbol{m}^{(l)}$, their joint probability distribution can be considered as a product of the beta distributions and defined as

$$\begin{aligned} \text{Beta}(\boldsymbol{m}^{(l)}; \alpha_l, \beta_l) &= \prod_i \text{Beta}(m_i^{(l)}; \alpha_l, \beta_l) \\ &= \prod_i \frac{(m_i^{(l)})^{\alpha_l - 1}(1 - m_i^{(l)})^{\beta_l - 1}}{\int_0^1 u^{\alpha_l - 1}(1-u)^{\beta_l - 1}\, du}, \end{aligned} \quad (1)$$

where the shape parameters α_l and β_l are greater than 0, $l = 1, \cdots, L$.

The probability density functions (PDFs) of beta distribution are exemplified in Figure 2. The beta distribution allows the random variable falling in the bounded interval $[0, 1]$, which allows the neural network to drop each node in various levels. The soft dropout can approximate the standard dropout [17] when $\alpha, \beta \to 0$. In addition, the soft dropout with $\alpha, \beta = 1$ can be considered as a dropout with uniform noise.

2.2. Stochastic Gradient Variational Bayes (SGVB) for Adaptive Soft Dropout

In this section, we introduce the adaptive soft dropout trained by the stochastic gradient variational Bayes (SGVB) [26] algorithm.

We define a dataset $\boldsymbol{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$ and a dropout-masked model parameter set $\boldsymbol{W} = \{\boldsymbol{w}_l\}_{l=1}^{L}$ and obtain the joint distribution of $\boldsymbol{D}$ and $\boldsymbol{W}$ as

$$p(\boldsymbol{D}, \boldsymbol{W}) = p(\boldsymbol{D}|\boldsymbol{W})p(\boldsymbol{W}). \tag{2}$$

Following [26], we introduce $q_{\boldsymbol{\Phi}}(\boldsymbol{W})$ as the approximated distribution of $\boldsymbol{W}$ with parameter $\boldsymbol{\Phi} = \{\boldsymbol{\Theta}, \boldsymbol{\Lambda}\}$ where $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_1, \cdots, \boldsymbol{\lambda}_L\}$ is a set of distribution parameters for each layer and consider the expectation of the joint distribution in (2) *w.r.t.* $q_{\boldsymbol{\Phi}}(\boldsymbol{W})$ as

$$\begin{aligned} &\underbrace{\int q_{\boldsymbol{\Phi}}(\boldsymbol{W}) \log \frac{p(\boldsymbol{D}, \boldsymbol{W})}{q_{\boldsymbol{\Phi}}(\boldsymbol{W})} \, d\boldsymbol{W}}_{L(\boldsymbol{\Phi})} \\ &= \underbrace{\int q_{\boldsymbol{\Phi}}(\boldsymbol{W}) \log p(\boldsymbol{D}|\boldsymbol{W}) \, d\boldsymbol{W}}_{L_D(\boldsymbol{\Phi})} - \underbrace{\int q_{\boldsymbol{\Phi}}(\boldsymbol{W}) \log \frac{q_{\boldsymbol{\Phi}}(\boldsymbol{W})}{p(\boldsymbol{W})} \, d\boldsymbol{W}}_{\mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\Phi}}(\boldsymbol{W})||p(\boldsymbol{W}))}, \end{aligned} \tag{3}$$

where $L(\boldsymbol{\Phi})$ is the lower bound of $\mathrm{E}_{\boldsymbol{\Phi}}\left[p(\boldsymbol{W}|\boldsymbol{D})\right]$ in variational inference, $L_D(\boldsymbol{\Phi})$ is the expected log-likelihood, and D_{KL} means the Kullback-Leibler (KL) divergence.

Instead of maximizing the lower bound $L(\boldsymbol{\Phi})$, we can maximize the right hand side (RHS) of (3), which is the expected log-likelihood $L_D(\boldsymbol{\Phi})$ subtracting the KL divergence between the approximated distribution $q_{\boldsymbol{\Phi}}(\boldsymbol{W})$ and the prior distribution $p(\boldsymbol{W})$.

Here, $L_D(\boldsymbol{\Phi})$ can be approximated by the expected log-likelihood $L_D^{\mathrm{SGVB}}(\boldsymbol{\Phi})$ in SGVB when applying mini-batch stochastic gradient descent (SGD) algorithm in the training steps as

$$\begin{aligned} L_D(\boldsymbol{\Phi}) &= \sum_{\boldsymbol{x}\in\boldsymbol{X}, \boldsymbol{y}\in\boldsymbol{Y}} \mathrm{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{W})}[\log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W})] \\ &\approx L_D^{\mathrm{SGVB}}(\boldsymbol{\Phi}) = \frac{N}{M} \sum_{i=1}^{M} \log p(\boldsymbol{y}^i|\boldsymbol{x}^i, \boldsymbol{W} = f(\boldsymbol{\epsilon}; \boldsymbol{\Phi})), \end{aligned} \tag{4}$$

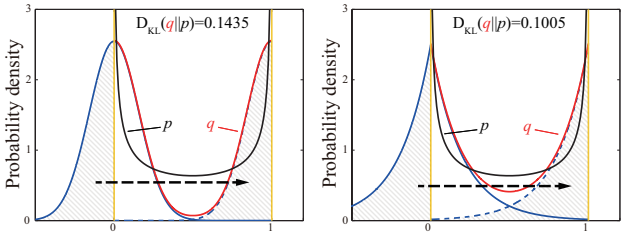
where N and M are the data point number in $\boldsymbol{D}$ and the batch size in the training steps, respectively. $\boldsymbol{y}^i$ and $\boldsymbol{x}^i$ are the target and the input of the i^{th} samples in the minibatches. $\boldsymbol{W}$ can be calculated by random samples $\boldsymbol{\epsilon}$ and a differentiable function $f(\cdot; \boldsymbol{\Phi})$).

For the l^{th} layer of a neural network, we define $\boldsymbol{x}_{l-1} \in R^{K_{l-1}}$ and $\boldsymbol{z}_l \in R^{K_l}$ as the input and the output of the layer and can obtain the layer function with standard dropout mask $\boldsymbol{m}_l$ as

$$\boldsymbol{z}_l = a\left(\boldsymbol{m}_l \odot (\boldsymbol{\theta}_l^{\mathrm{T}} \boldsymbol{x}_l)\right), \tag{5}$$

where $a(\cdot)$ is the activation function and $\odot$ is the element-wise multiplication.

The basic principle in SGVB [26] is to parameterize the dropout-masked parameter matrix $\boldsymbol{w}_l \sim q_{\boldsymbol{\theta}_l}(\boldsymbol{w}_l)$ by $\boldsymbol{w}_l = f(\boldsymbol{\epsilon}_l; \boldsymbol{\theta}_l, \boldsymbol{\lambda}_l)$ where $f(\cdot)$ is the differentiable function introduced in (4) and $\epsilon_{li} \sim p(\epsilon_{li}), i = 1, \cdots, K_l$ in $\boldsymbol{\epsilon}_l$ is a random noise variable following a prior distribution with fixed parameter(s). The layer function with the SGVB-based $\boldsymbol{w}_l$ can be defined as



(a) Gaussian approximation (b) Laplace approximation

Fig. 3: Illustration of beta approximation via (a) Gaussian distribution and (b) Laplace distribution. The black solid line are the beta PDFs p with $\alpha = \beta = 0.5$, the blue solid lines are the Gaussian PDF in (a) and the Laplace PDF in (b) respectively, the blue dashed lines are shifted from the corresponding negative zones of the PDFs, and the red solid lines are the approximated PDFs q.

$$\boldsymbol{z}_l = a(\boldsymbol{w}_l^{\mathrm{T}} \boldsymbol{x}_l) = a(f(\boldsymbol{\epsilon}_l; \boldsymbol{\theta}_l, \boldsymbol{\lambda}_l)^{\mathrm{T}} \boldsymbol{x}_l). \tag{6}$$

Although the beta PDF is differentiable in the interval of $(0, 1)$, the beta distribution is unfeasible to be directly extended in SGVB. In other words, it is difficult to find an easy and practicable representation for a beta variable by the differentiable function $f(\cdot)$ and the random variables $\boldsymbol{\epsilon}$ discussed above. However, we can approximate the beta distribution via a two half-Gaussian or two half-Laplace distributions which can be obtained from a single Gaussian or a single Laplace distribution respectively as shown in Figure 3. When applying the former, we can obtain the differentiable function $f(\boldsymbol{\epsilon}_l; \boldsymbol{\theta}_l, \sigma_1^{(l)}, \sigma_2^{(l)})$ of the l^{th} layer as

$$\begin{aligned} &f(\boldsymbol{\epsilon}_l; \boldsymbol{\theta}_l, \sigma_1^{(l)}, \sigma_2^{(l)}) = \boldsymbol{\theta}_l \cdot \mathrm{diag}(\max(0, \min(1, \boldsymbol{\zeta}_l))), \\ &\boldsymbol{\zeta}_l = \boldsymbol{\tau}_l \odot (\boldsymbol{\epsilon}_l \sigma_1^{(l)}) + (1 - \boldsymbol{\tau}_l) \odot (1 + \boldsymbol{\epsilon}_l \sigma_2^{(l)}), \end{aligned} \tag{7}$$

where $\epsilon_{li} \sim \mathcal{N}(0, 1), i = 1, \cdots, K_l$, $\sigma_1^{(l)}$ and $\sigma_2^{(l)}$ are scale parameters of two half-Gaussian distributions respectively, (*i.e.*, $\boldsymbol{\lambda}_l = \{\sigma_1^{(l)}, \sigma_2^{(l)}\}$,) $\mathrm{diag}(\cdot)$ is the matrix operation that transforms a vector into a square diagonal matrix with the vector as the main diagonal, and

$$\tau_{li} = \begin{cases} 1, \epsilon_{li} \geq 0 \\ 0, \epsilon_{li} < 0 \end{cases}. \tag{8}$$

Similarly, when adopting the latter, the intermediate vector $\boldsymbol{\zeta}_l$ can be presented as

$$\boldsymbol{\zeta}_l = \boldsymbol{\tau}_l \odot (\boldsymbol{\epsilon}_l b_1^{(l)}) + (1 - \boldsymbol{\tau}_l) \odot (1 + \boldsymbol{\epsilon}_l b_2^{(l)}), \tag{9}$$

where $\epsilon_{li} \sim \mathrm{Laplace}(0, 1)$, and $b_1^{(l)}$ and $b_2^{(l)}$ are the scale parameters of two half-Laplace distributions respectively, *i.e.*, $\boldsymbol{\lambda}_l = \{b_1^{(l)}, b_2^{(l)}\}$.

Table 1: Test accuracies (%) on the MNIST dataset. 1-hidden-layer FC neural networks are constructed with different hidden node numbers. Our soft dropout techniques (with various parameter settings and with Gaussian and Laplace approximations) are compared with the referred methods. Note that the best results of each FC neural network structure are marked in **bold** fonts, respectively.

Hidden node number		100	500	1000
No dropout		96.98 ± 0.09	97.27 ± 0.06	97.25 ± 0.11
Fixed dropout rate	Dropout, Bernoulli ($p = 0.5$)	96.65 ± 0.17	98.09 ± 0.03	98.23 ± 0.08
	Dropout, Gaussian ($p = 0.5$)	95.02 ± 0.37	98.11 ± 0.10	98.13 ± 0.09
	MC dropout ($p = 0.5$)	87.24 ± 0.84	96.76 ± 0.17	97.35 ± 0.07
	Soft dropout ($\alpha = 0.1$, $\beta = 0.1$)	97.15 ± 0.12	98.20 ± 0.06	98.35 ± 0.11
	Soft dropout ($\alpha = 0.5$, $\beta = 0.5$)	97.71 ± 0.09	98.35 ± 0.07	98.41 ± 0.04
	Soft dropout ($\alpha = 0.9$, $\beta = 0.9$)	$\mathbf{98.06 \pm 0.19}$	$\mathbf{98.51 \pm 0.05}$	$\mathbf{98.51 \pm 0.05}$
	Soft dropout ($\alpha = 1.0$, $\beta = 1.0$)	97.60 ± 0.14	98.18 ± 0.10	98.23 ± 0.12
Adaptive dropout rate	Concrete dropout	97.85 ± 0.08	98.26 ± 0.10	98.29 ± 0.07
	Variational dropout	98.09 ± 0.06	98.34 ± 0.07	98.42 ± 0.07
	Gaussian soft dropout	98.15 ± 0.06	$\mathbf{98.49 \pm 0.18}$	$\mathbf{98.56 \pm 0.16}$
	Laplace soft dropout	$\mathbf{98.28 \pm 0.06}$	98.45 ± 0.06	98.44 ± 0.03

3. EXPERIMENTAL RESULTS AND DISCUSSIONS

In this section, we conduct experiments on both the classification and the regression tasks to demonstrate the effectiveness of the proposed soft dropout method in preventing overfitting. We show the experimental results of the proposed soft dropout with different settings and its adaptive dropout rate versions compared with the standard dropout with Bernoulli noise [17] and Gaussian noise [18], the MC dropout [23], the concrete dropout [24], and the variational dropout [14]. As for the classification task, we evaluate the methods on both the MNIST [27] and the CIFAR-10 [28] datasets in Section 3.1. In addition, we evaluate them on the UCI datasets [29] for the regression task in Section 3.2.

3.1. Classification on MNIST and CIFAR-10 datasets

Experimental results on the standard classification benchmark, *i.e.*, MNIST dataset using 1-hidden-layer FC neural networks are shown in Table 1. To evaluate behaviour of the methods in relation to different model sizes, *i.e.*, hidden node number, we conducted several FC neural networks with 100, 500, and 1000 hidden nodes, respectively. Except for the concrete dropout and the variational dropout which have adaptive dropout rates, other referred methods have their dropout rate $1 - p = 0.5$ in the training steps. Meanwhile, we set $\alpha = \beta$, as discussed in Section 2.1, with different values at 0.1, 0.5, 0.9, and 1.0 for soft dropout with a fixed dropout rate and two adaptive soft dropout methods are named as the Gaussian soft dropout and the Laplace soft dropout, respectively. Adopting the SGD algorithm, we trained each method over 100 epochs to guarantee its convergence. All the methods were experimented 5 runs with random initialization to compare the means and standard deviations of the classification accuracies.

From Table 1, for the fixed rate part, the proposed soft dropout method with $\alpha = \beta = 0.9$ achieves the best accuracies of 98.06% on a 1-hidden-layer FC neural network with 100 hidden nodes compared with different referred methods, although those with $\alpha = \beta = 0.1$, $\alpha = \beta = 0.5$, and $\alpha = \beta = 1.0$ also obtain competitive results. Moreover, we achieve 98.51%, which is the best accuracy on the proposed soft dropout at $\alpha = \beta = 0.9$, using both 500- and 1000-hidden-node FC neural networks. Meanwhile, compared with two adaptive-rate dropout versions, the Laplace soft dropout performs the best at 98.28% on the 100-hidden-node FC neural network, while the Gaussian soft dropout achieves the best accuracies on the 500- and 1000-hidden-node FC neural networks at 98.49% and 98.56%, respectively.

We then evaluated the performance of the proposed soft dropout with the CIFAR-10 dataset. We conducted a CNN for image classification with a scale parameter k to adjust the model size. The CNN has two convolutional layers with $32 \times k$ and $64 \times k$ channels followed by two FC layers with $128 \times k$ hidden nodes of each. In the experiment settings, we apply the scale parameter k at 1, 2, and 3 respectively. We also set $p = 0.5$ for the standard dropout with Bernoulli noise [17] and Gaussian noise [18], and the MC dropout [23], while α of the soft dropout is set as 0.1, 0.5, 0.9, and 1.0, respectively, equal to β. All the models have been trained for over 100 epochs using the SGD algorithm, and each experiment is run 5 times with random initial settings.

Table 2 shows the comparisons of the classification accuracies on test set of the CIFAR-10 dataset among distinct referred methods and the proposed soft dropout method with different parameter settings and both adaptive versions. Our soft dropout with $\alpha = \beta = 0.1$ achieves the best accuracy of 78.92%, which is significantly higher than the MC dropout and the concrete dropout, and slightly higher than the stan-

Table 2: Test accuracies (%) on the CIFAR-10 dataset. CNNs are constructed with different model sizes. Our soft dropout techniques (with various parameter settings and with Gaussian and Laplace approximations) are compared with the referred methods. Note that the best results of each CNN structure are marked in **bold** fonts, respectively.

k		1	2	3
No dropout		73.40 ± 0.42	75.24 ± 0.22	75.72 ± 0.13
Fixed dropout rate	Dropout, Bernoulli ($p = 0.5$)	78.55 ± 0.49	79.50 ± 0.21	79.68 ± 0.11
	Dropout, Gaussian ($p = 0.5$)	78.78 ± 0.36	79.49 ± 0.26	79.62 ± 0.30
	MC dropout ($p = 0.5$)	72.23 ± 0.96	77.47 ± 0.39	79.32 ± 0.17
	Soft dropout ($\alpha = 0.1, \beta = 0.1$)	$\mathbf{78.92 \pm 0.55}$	$\mathbf{80.18 \pm 0.24}$	$\mathbf{80.18 \pm 0.18}$
	Soft dropout ($\alpha = 0.5, \beta = 0.5$)	77.53 ± 0.39	78.99 ± 0.27	79.10 ± 0.27
	Soft dropout ($\alpha = 0.9, \beta = 0.9$)	76.34 ± 0.38	77.93 ± 0.23	78.30 ± 0.20
	Soft dropout ($\alpha = 1.0, \beta = 1.0$)	76.29 ± 0.16	77.82 ± 0.29	78.28 ± 0.43
Adaptive dropout rate	Concrete dropout	72.55 ± 0.97	75.10 ± 0.23	75.04 ± 1.09
	Variational dropout	$\mathbf{77.73 \pm 0.26}$	$\mathbf{78.61 \pm 0.16}$	78.43 ± 0.32
	Gaussian soft dropout	77.18 ± 0.14	78.41 ± 0.10	$\mathbf{78.79 \pm 0.09}$
	Laplace soft dropout	77.31 ± 0.32	78.54 ± 0.24	78.75 ± 0.19

Table 3: Test RMSEs on the four UCI datasets and corresponding input feature sizes. Our soft dropout techniques (with various parameter settings and with Gaussian and Laplace approximations) are compared with the referred methods. Note that the best results of each dataset are marked in **bold** fonts, respectively.

Dataset Input size		Boston Housing 13	Concrete Strength 8	Wine Quality Red 11	Yacht Hydrodynamics 6
No dropout		8.55 ± 0.01	15.82 ± 0.03	0.8154 ± 0.0014	13.17 ± 0.18
Fixed dropout rate	Dropout, Bernoulli ($p = 0.5$)	8.51 ± 0.01	15.78 ± 0.01	0.8130 ± 0.0005	13.02 ± 0.07
	Dropout, Gaussian ($p = 0.5$)	8.51 ± 0.01	$\mathbf{15.78 \pm 0.01}$	$\mathbf{0.8128 \pm 0.0003}$	12.97 ± 0.16
	MC dropout ($p = 0.5$)	8.99 ± 0.30	15.82 ± 0.10	0.8404 ± 0.0056	12.50 ± 0.26
	Soft dropout ($\alpha = 0.1, \beta = 0.1$)	$\mathbf{8.51 \pm 0.01}$	15.79 ± 0.02	0.8131 ± 0.0003	$\mathbf{12.45 \pm 0.36}$
	Soft dropout ($\alpha = 0.5, \beta = 0.5$)	8.53 ± 0.02	15.78 ± 0.02	0.8134 ± 0.0003	12.50 ± 0.40
	Soft dropout ($\alpha = 0.9, \beta = 0.9$)	8.52 ± 0.01	15.79 ± 0.02	0.8136 ± 0.0003	12.83 ± 0.33
	Soft dropout ($\alpha = 1.0, \beta = 1.0$)	8.54 ± 0.01	15.81 ± 0.02	0.8134 ± 0.0004	13.04 ± 0.08
Adaptive dropout rate	Concrete dropout	8.77 ± 0.09	16.03 ± 0.03	0.8273 ± 0.0049	13.25 ± 0.33
	Variational dropout	8.51 ± 0.01	15.79 ± 0.02	0.8133 ± 0.0004	13.06 ± 0.06
	Gaussian soft dropout	8.49 ± 0.01	15.76 ± 0.01	$\mathbf{0.8128 \pm 0.0003}$	12.32 ± 0.04
	Laplace soft dropout	$\mathbf{8.47 \pm 0.01}$	$\mathbf{15.75 \pm 0.02}$	0.8133 ± 0.0005	$\mathbf{12.28 \pm 0.02}$

dard dropout when $k = 1$. For both of the other two model sizes (*i.e.*, $k = 2$ and 3), the proposed soft dropout with $\alpha = \beta = 0.1$ obtains the best accuracies of 80.18% as well. With larger α and β of the soft dropout, the classification accuracy drops markedly. In addition, both the Gaussian soft dropout and the Laplace soft dropout perform competitively among the variational dropout with $k = 1, 2$, while the Gaussian soft dropout obtains the best performance (78.79%) when $k = 3$.

3.2. Regression on UCI datasets

We further conducted experiments in a regression setting using the well-known UCI datasets. All the methods above were tested using a FC neural network with 2 hidden layers, 50 hidden nodes each. The root mean squared error (RMSE) is considered as the metric for all the methods, which is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}(\boldsymbol{y} - \hat{\boldsymbol{y}})^{\mathrm{T}}(\boldsymbol{y} - \hat{\boldsymbol{y}})}, \tag{10}$$

where $\boldsymbol{y}$ is the target, $\hat{\boldsymbol{y}}$ is the output of the FC neural network, and N is the number of test points. All the methods were trained for 200 epochs using the SGD algorithm with 5 runs under random initialization.

Regression results on the test sets are reported in Table 3. The proposed soft dropout with $\alpha = \beta = 0.1$ achieves best performance among other dropout techniques with the fixed dropout rate on the Boston Housing and the Yacht Hydrodynamics datasets with mean RMSE at 8.51 and 12.45, respectively. At the meantime, the standard dropout with Gaussian noise has the lowest mean of RMSE at 15.78 and 0.8128 on the Concrete Strength and the Wine Quality Red datasets, re-

spectively. The Laplace soft dropout outperforms all the other methods in terms of mean RMSE on the Boston Housing, the Concrete Strength, and the Yacht Hydrodynamics datasets, while the Gaussian soft dropout performs best on the Wine Quality Red dataset.

4. CONCLUSIONS

In this paper, we proposed the soft dropout technique with random "soft" masks following beta distribution to prevent overfitting. Comparing with the "hard" dropout with Bernoulli masks, the soft masks have continuous values in the interval of $[0, 1]$ which can express the "drop" in various levels. Meanwhile, in order to implement the adaptive dropout rate via adaptive distribution parameters learned by the variational Bayes optimization, we utilized the half-Gaussian distribution and the half-Laplace distribution to approximate the beta prior of random masks, separately, and applied the stochastic gradient variational Bayes (SGVB) algorithm for optimizing the distribution parameters. Compared with the standard soft dropout with fixed dropout rate, the adaptive soft dropout methods called Gaussian and Laplace soft dropout obtain performance improvement on most datasets. In addition, the proposed soft dropout and its adaptive versions achieve performance improvement, comparing with the five referred methods on both the image classification and the regression tasks. In practical application of neural network training, the Laplace soft dropout is recommended due to its smaller KL divergence between approximated and actual beta distributions than the Gaussian soft dropout as shown in Figure 3, though they have no significant difference in performance.

5. REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] G. Huang, Z. Liu, L. V. Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269.

[4] X. Li, L. Yu, D. Chang, Z. Ma, and J. Cao, "Dual cross-entropy loss for small-sample fine-grained vehicle classification," *IEEE Transactions on Vehicular Technology*, 2019.

[5] X. Li, D. Chang, T. Tian, and J. Cao, "Large-margin regularized softmax cross-entropy loss," *IEEE Access*, vol. 7, pp. 19572–19578, 2019.

[6] Z. Ma, Y. Ding, S. Wen, J. Xie, Y. Jin, Z. Si, and H. Wang, "Shoe-print image retrieval with multi-part weighted cnn," *IEEE ACCESS*, vol. 7, pp. 59728 – 59736, 2019.

[7] P. Xu, Y. Huang, T. Yuan, K. Pang, Y. Z. Song, T. Xiang, T. M. Hospedales, Z. Ma, and J. Guo, "Sketchmate: Deep hashing for million-scale human sketch retrieval," in *Computer Vision and Pattern Recognition*, 2018, pp. 8090–8098.

[8] P. Xu, Q. Yin, Y. Huang, Y. Z. Song, Z. Ma, L. Wang, T. Xiang, W. B. Kleijn, and J. Guo, "Cross-modal subspace learning for fine-grained sketch-based image retrieval," *Neurocomputing*, vol. 278, pp. 75–86, 2018.

[9] P. Xu, Q. Yin, Y. Qi, Y. Z. Song, Z. Ma, L. Wang, and J. Guo, "Instance-level coupled subspace learning for fine-grained sketch-based image retrieval," in *European Conference on Computer Vision Workshops*. Springer, 2016, pp. 19–34.

[10] Z. Ma, H. Yu, W. Chen, and J. Guo, "Short utterance based speech language identification in intelligent vehicles with time-scale modifications and deep bottleneck features," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 121–128, 2019.

[11] H. Yu, Z. H. Tan, Z. Ma, R. Martin, and J. Guo, "Spoofing detection in automatic speaker verification systems using dnn classifiers and dynamic acoustic features," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4633–4644, 2018.

[12] Z. Ma, J. Xie, H. Li, Q. Sun, F. Wallin, Z. Si, and J. Guo, "Deep neural network-based impacts analysis of multimodal factors on heat demand prediction," *IEEE Transactions on Big Data*, pp. 1–1, 2019.

[13] J. Xie, J. Guo, Z. Ma, J. H. Xue, Q. Sun, H. Li, and J. Guo, "Sea: A combined model for heat demand prediction," in *IEEE International Conference on Network Infrastructure and Digital Content*, Aug 2018, pp. 71–75.

[14] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Neural Information Processing Systems*, 2015, vol. 28, pp. 2575–2583.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[16] Y. Tokozume, Y. Ushiku, and T. Harada, "Between-class learning for image classification," in *Computer Vision and Pattern Recognition*, 2018, pp. 5486–5494.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv: Neural and Evolutionary Computing*, 2012.

[18] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[19] R. Tibshirani, "Regression shrinkage and selection via the lasso: a retrospective," *Journal of The Royal Statistical Society Series B-statistical Methodology*, vol. 73, pp. 273–282, 2011.

[20] A. Labach and H. Salehinejad, "Survey of dropout methods for deep neural networks," *arXiv preprint arXiv:1904.13310*, pp. 1–11, 2019.

[21] S. I. Wang and C. D. Manning, "Fast dropout training," in *International Conference on Machine Learning*, 2013, pp. 118–126.

[22] S. Maeda, "A bayesian encourages dropout," in *International Conference on Learning Representation*, 2015.

[23] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: representing model uncertainty in deep learning," in *International Conference on Machine Learning*, 2016, pp. 1050–1059.

[24] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Neural Information Processing Systems*, 2017, pp. 3581–3590.

[25] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Neural Information Processing Systems*, 2016, pp. 1027–1035.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014.

[27] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 1998.

[28] A. Krizhevsky, "Learning multiple layers of features from tiny images," techreport, CIFAR, 2009.

[29] D. Dua and C. Graff, "UCI machine learning repository," http://archive.ics.uci.edu/ml, 2017.