Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Shakeout: A New Regularized Deep Neural Network Training Scheme

Guoliang Kang, Jun Li, Dacheng Tao
Centre for Quantum Computation and Intelligent Systems
Faculty of Engineering and Information Technology, University of Technology Sydney
{Guoliang.Kang@student, Jun.Li@, Dacheng.Tao@}uts.edu.au

Abstract

Recent years have witnessed the success of deep neural networks in dealing with a plenty of practical problems. The invention of effective training techniques largely contributes to this success. The so-called "Dropout" training scheme is one of the most powerful tool to reduce over-fitting. From the statistic point of view, Dropout works by implicitly imposing an L_2 regularizer on the weights. In this paper, we present a new training scheme: Shakeout. Instead of randomly discarding units as Dropout does at the training stage, our method randomly chooses to enhance or inverse the contributions of each unit to the next layer. We show that our scheme leads to a combination of L_1 regularization and L_2 regularization imposed on the weights, which has been proved effective by the Elastic Net models in practice. We have empirically evaluated the Shakeout scheme and demonstrated that sparse network weights are obtained via Shakeout training. Our classification experiments on real-life image datasets MNIST and CIFAR-10 show that Shakeout deals with over-fitting effectively.

Introduction

Deep neural network has recently achieved impressive success in a number of machine learning and pattern recognition tasks and been under intensive research. Hierarchical neural network has been known for decades, and there are a number of essential factors contributing to its recent rising, such as large volume of data and powerful computational resources. However, arguably the most important contributor to the success of deep neural network is the discovery of efficient training methods (Hinton, Osindero, and Teh 2006; Bengio 2009; Bengio, Courville, and Vincent 2013; Vincent et al. 2008; 2010).

A particular interesting advance in this strand of research is the invention of the Dropout training scheme (Hinton et al. 2012). At the operational level, Dropout adjusts the network evaluation step (feed-forward) at the training stage, where a portion of units are randomly discarded. The empirical effect of this seemingly trivial trick is impressive. Dropout enhances the generalization performance of neural networks considerably, and is behind many record-holders of widely recognized benchmarks (Krizhevsky, Sutskever, and Hinton 2012; Lin, Chen, and Yan 2013; Szegedy et al. 2014). The intriguing effect has drawn great attention from the research community, and found applications in wider range of problems (Chen et al. 2014). Theoretical research from the viewpoint of statistical learning has pointed out the connections between the Dropout operations and model regularization, which is the *de facto* recipe of reducing over-fitting for complex models in practical machine learning. For example, Wager et al. (Wager, Wang, and Liang 2013) have shown that Dropout implicitly imposes L_2 regularization on the network weights.

In this paper, we propose a new regularized deep neural network training scheme: Shakeout. It is straightforward and simple to implement: instead of randomly discarding units as Dropout does at the training stage, our method randomly chooses to enhance or inverse the contributions of each unit to the next layer. These operations can lead to a profound change to the regularization effect. We have proved that the new training scheme imposes a combination of L_1 regularization and L_2 regularization on the weights of the model. We call our scheme "Shakeout" because of the randomly "shaking" process and the L_1 regularization effect pushing network weights to zero. The combination of L_1 and L_2 regularization on the model parameters has been shown effective in statistical learning: the Elastic Net (Zou and Hastie 2005; Zhou, Tao, and Wu 2011) has the desirable properties of producing sparse models which generalize well and have good interpretability.

Our Shakeout scheme has been applied to training deep neural networks for image classification. The scheme can produce models with expected statistical attributes. Those models generalize better than the standardly trained networks. In adverse training conditions, such as with scarce data, the new training scheme outperforms Dropout training.

The organization of this paper is as follows. After reviewing related background in the next section, we introduce our Shakeout scheme in a simplified context of training generalized linear models. We then apply Shakeout to train multilayer neural networks. In the experiment section, we show that Shakeout can produce sparse neural network weights. Moreover we demonstrate the results of classification experiments on MNIST and CIFAR-10. Finally, our work is concluded.

Related Works

Multilayer artificial neural network has long been known as a powerful computational model (Williams and Hinton 1986) for pattern recognition. However, in the early days, the applications of deep neural networks were mostly limited except for the success obtained in some specialized domains (LeCun et al. 1998). It is expected that the representative power of the network will become stronger as the architecture gets deeper (Bengio 2009), while training a deep neural network is a non-trivial task because of some difficult training issues (Bengio 2009). The rise of deep learning recently can be largely attributed to more effective training methods (Hinton, Osindero, and Teh 2006; Bengio, Courville, and Vincent 2013; Vincent et al. 2010; 2008). The reason for the good performance of these training techniques can be ascribed to the induced regularization effect. This kind of explanation is supported by quite a few existing empirical and theoretical studies (Erhan et al. 2010; Warde-Farley et al. 2013; Wager, Wang, and Liang 2013).

Dropout was first proposed by Hinton (Hinton et al. 2012) to reduce over-fitting. Many evidences show that Dropout may work because of its good approximation to model averaging and imposing regularization on the networks. Srivastava (Srivastava et al. 2014) and Warde-Farley (Wager, Wang, and Liang 2013) exhibited through experiments that weight scaling approximation is an accurate alternative for geometric mean over all possible sub-networks. Dropout can be regarded as a way of adding noise into the neural network. By marginalizing the noise, Srivastava (Srivastava et al. 2014) proved for linear regression model that the deterministic version of Dropout is equivalent to adding an adaptive L_2 regularizer on the weights. Furthermore, Wager (Wager, Wang, and Liang 2013) extended the conclusion to generalized linear models (GLMs) using a quadratic approximation to the noising penalty.

Shakeout Training

We first illustrate our Shakeout scheme for the training of generalized linear models (GLMs). We then move on to the more complex and practical scenario of training multilayer neural networks which can be viewed as a hierarchical construction of GLMs.

Shakeout for Generalized Linear Models

Let the input be represented as a set of features $\mathbf{x} = [x_1, x_2, \ldots, x_P]^\top$, a GLM stochastically characterizes a target y using the exponential family distributions $p(y|\mathbf{x}, \mathbf{w}) = h(y)\exp\{y\eta - A(\eta)\}$ where $\eta = \langle \mathbf{w}, \mathbf{x} \rangle$. The $h(y)$ is a quantity which is independent of $\mathbf{x}$ and $\mathbf{w}$. A is the log-partition function. The $\mathbf{w}$ is typically learned from a set of independent and identically distributed (i.i.d.) data by minimizing the negative log-likelihood loss function $\ell(\mathbf{w}; \mathbf{X}, \mathbf{y}) = -\log\{p(\mathbf{y}|\mathbf{X}, \mathbf{w})\}$: $\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w}} \ell(\mathbf{w}; \mathbf{X}, \mathbf{y})$ so that the observed data is the most plausible under the learned model.

The procedure of Shakeout operations is as follows(which applies for each training example, and we omit the example subscripts for clearer math symbols):

Step 1: Generate $\mathbf{r} = [r_1, r_2, \cdots, r_P]^\top$. First, sample $\hat{r}_j$ independently from Bernoulli$(1-\tau)$, $\tau \in [0, 1]$. Then, $r_j \leftarrow \hat{r}_j/(1-\tau), \forall j \in \{1, 2, \cdots, P\}$.

Step 2: $\widetilde{\eta} \leftarrow \sum_{j=1}^{P} x_j\{r_j(w_j + cs_j) - cs_j\}$ where $s_j = \text{sgn}(w_j), \forall j \in \{1, 2, \cdots, P\}$. $\text{sgn}(\cdot)$ is a signum function that extracts the sign of a real number. c is a constant and $c \in \mathbb{R}, c > 0$.

The Shakeout operation is indeed to alter each input feature to a noisy version, where the noise is injected by a random "shaking" process. Formally, let $\widetilde{x}_j$ represent the noisy version of feature x_j. $\widetilde{\eta}$ in Step 2 can be written as $\widetilde{\eta} = \sum_{j=1}^{P} \widetilde{x}_j w_j$, where $\widetilde{x}_j = \epsilon_j x_j$ with $\epsilon_j = r_j + c(r_j - 1)/|w_j|$. Then

$$\epsilon_j = \begin{cases} \frac{1}{1-\tau} + c\frac{\tau}{(1-\tau)|w_j|} & r_j = 1/(1-\tau) \\ -c/|w_j| & r_j = 0 \end{cases} \tag{1}$$

Shakeout randomly chooses to enhance (if $r_j \leftarrow 1/(1-\tau)$, $\epsilon_j > 1/(1-\tau)$) or inverse (if $r_j \leftarrow 0$, $\epsilon_j < 0$) the contribution of original feature x_j to $\widetilde{\eta}$. However, in expectation, $\mathbb{E}_{r_j}[\widetilde{x}_j] = x_j$.

It has been pointed out by a number of studies that by carefully injecting artificial noises into the input features, taking expectation of the original loss function over the noises will lead to an altered deterministic one with an additional regularization term (Wager, Wang, and Liang 2013; Rifai et al. 2011; Bishop 1995), i.e. $\mathbb{E}_{noise}[\ell(\mathbf{w}; \widetilde{\mathbf{X}}, \mathbf{y})] = \ell(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \pi(\mathbf{w})$; where $\widetilde{\mathbf{X}}$ consists of the noisy feature vectors. The regularization term $\pi(\mathbf{w})$ is determined by the characteristics of the noise and is independent of the target $\mathbf{y}$. For example, Wager et al. (Wager, Wang, and Liang 2013) showed that the Dropout scheme, corresponding to inducing blackout noise $\widetilde{x}_j \leftarrow x_j r_j$ helps introduce a L_2 regularizer $\pi(\mathbf{w})$. From the viewpoint of regularizing the training objective, we can show that the feature noise injected by Shakeout implicitly induces L_1 regularization and L_2 regularization on $\mathbf{w}$.

Theorem 1. *For the generalized linear models, in expectation, Shakeout is equal to introducing a combination of L_1 regularization and L_2 regularization on the weights* $\mathbf{w}$.

Proof. Let $\widetilde{\ell}(\mathbf{w}; \mathbf{X}, \mathbf{y})$ denote the loss function for Shakeout scheme, which equals to the expectation of the original loss function over noises, i.e. $\widetilde{\ell}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{N} \mathbb{E}_{\mathbf{r}_i} \ell(\mathbf{w}; \widetilde{\mathbf{x}}_i, y_i)$. Note that $\mathbb{E}_{\mathbf{r}_i}\widetilde{\eta}_i = \eta_i = \mathbf{w}^\top \mathbf{x}_i$, then we can easily get

$$\widetilde{\ell}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \ell(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \pi(\mathbf{w}) \tag{2}$$

where

$$\pi(\mathbf{w}) = \sum\nolimits_{i=1}^{N} \mathbb{E}_{\mathbf{r}_i}[A(\widetilde{\eta}_i) - A(\eta_i)]$$

Using quadratic approximation, $A(\widetilde{\eta}_i) - A(\eta_i) \approx A^{'}(\eta_i)(\widetilde{\eta}_i - \eta_i) + \frac{1}{2}A^{''}(\eta_i)(\widetilde{\eta}_i - \eta_i)^2$, then

$$\pi(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} A^{''}(\eta_i) Var(\widetilde{\eta}_i) \tag{3}$$

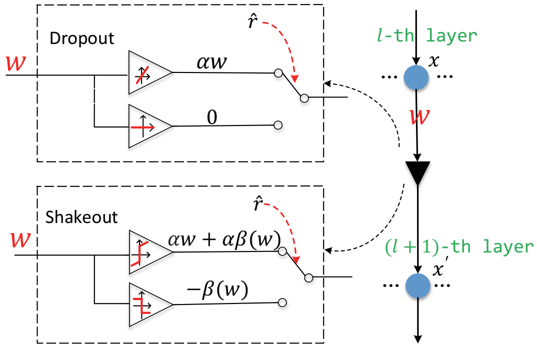

Figure 1: Shakeout and Dropout in the feed-forward computation of a neural network. This figure shows how Shakeout and Dropout affect a unit x contributing to a higher layer unit x' in a neural network. In the original network, the connections are weighted by w. In Dropout and Shakeout, a random switch $\hat{r}$ controls how w is modified. The manipulation of w is illustrated within the amplifier icons (the red curves, best seen with colours). The coefficients are $\alpha = 1/(1-\tau)$ and $\beta(w) = c\tau s(w)$, where $s(w)$ extracts the sign of w and $c > 0$, $\tau \in [0, 1]$. Note the sign of $\beta(w)$ is always the same with that of w. See equations (1) and (5) for more details.

For each training example $\mathbf{x}_i$, the variation of $\widetilde{\eta}_i$ is

$$Var[\widetilde{\eta}_i] = \frac{\tau}{1-\tau}\sum_{j=1}^{P}(w_j^2 X_{ij}^2 + 2c|w_j|X_{ij}^2) + \Omega$$

The term $\Omega = \frac{\tau}{1-\tau}\sum_{j=1}^{P} X_{ij}^2 c^2$ has nothing to do with $\mathbf{w}$, so we can ignore this term. From (3), the penalty term can be written as

$$\pi(\mathbf{w}) = \frac{\tau}{2(1-\tau)}||\mathbf{\Gamma}\mathbf{w}||_2^2 + c\frac{\tau}{1-\tau}||\mathbf{\Gamma}^2\mathbf{w}||_1 \quad (4)$$

where $\mathbf{\Gamma}$ is a diagonal matrix with diagonal entries equal to the square roots of corresponding ones in $\mathbf{X}^T\mathbf{V}\mathbf{X}$. $\mathbf{V} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with diagonal entries $A''(\eta_i)$.

So from Eq. (4), we can see that for the generalized linear models, in expectation, Shakeout is equal to introducing regularization penalizing the L_1-norm and L_2-norm of the weights $\mathbf{w}$. □

In Shakeout scheme, hyper-parameters τ and c relate to the strength of regularization effect. In particular, when $\tau = 0$, the noise is eliminated and the model becomes a standard GLM. Moreover, the Dropout is the special case of Shakeout when $c = 0$, and a higher value of τ means a stronger L_2 regularization imposed on the weights. Generally, when τ is fixed ($\tau \neq 0$), a higher value of c means a stronger L_1 regularization effect imposed on the weights and is expected to lead to much sparser weights of the model. We will verify this property in our experiment section later.

Shakeout for Multilayer Neural Networks

The Shakeout is readily applicable to the training of multilayer neural networks. Specifically, units in one layer are taken as the input features during the feed-forward computation of the units in the next higher layer, and the Shakeout noises are introduced to the lower layer of units.

The Shakeout feed-forward computation is as follows

$$u_i^{(l+1)} = \sum_{j=1}^{P^{(l)}} x_j^{(l)}[r_j^{(l)}W_{ij}^{(l+1)} + c(r_j^{(l)} - 1)S_{ij}^{(l+1)}] + b_i^{(l+1)} \quad (5)$$

$$x_i^{(l+1)} = f(u_i^{(l+1)}) \quad (6)$$

where the superscript l represents the hidden layer of the network, and $i = 1, 2, \cdots, P^{(l+1)}$, $j = 1, 2, \cdots, P^{(l)}$. The output unit of layer l is represented by $x^{(l)}$. The connection between unit $x_j^{(l)}$ and unit $x_i^{(l+1)}$ is represented as $W_{ij}^{(l+1)}$. And the bias for unit i of layer $l+1$ is denoted by $b_i^{(l+1)}$. By extracting the signs of corresponding weights, $S_{ij}^{(l+1)}$ is set as $\mathrm{sgn}(W_{ij}^{(l+1)})$. After Shakeout operations, the linear combination $u_i^{(l+1)}$ of the noisy output units of layer l will be sent to the activation function $f(\cdot)$ to obtain the corresponding output of layer $l+1$, i.e. $x_i^{(l+1)}$.

During the back-propagation process, we should compute the gradients with respect to each unit in order to propagate the error. In Shakeout scheme, $\frac{\partial u_i^{(l+1)}}{\partial x_j^{(l)}}$ takes a new form

$$\frac{\partial u_i^{(l+1)}}{\partial x_j^{(l)}} = r_j^{(l)}(W_{ij}^{(l+1)} + cS_{ij}^{(l+1)}) - cS_{ij}^{(l+1)} \quad (7)$$

And the weights are updated following

$$\frac{\partial u_i^{(l+1)}}{\partial W_{ij}^{(l+1)}} = x_j^{(l)}[r_j^{(l)}(1 + c\frac{\partial S_{ij}^{(l+1)}}{\partial W_{ij}^{(l+1)}}) - c\frac{\partial S_{ij}^{(l+1)}}{\partial W_{ij}^{(l+1)}}] \quad (8)$$

where $S_{ij}^{(l)}$ is approximated by $\tanh$, for the convenience of computing the gradients.

The operations of Shakeout are illustrated in Fig. 1, with a comparison to those of Dropout.

It is worth noting that, the Shakeout operations are applied to the feed-forward process at the *training* stage for the beneficial effect of regularization, while at *test* stage, it is not necessary to disturb the evaluation of a given neural network with any noise, i.e. we adopt the resulting network without any Shakeout operation.

Experiments

In this section, we report empirical evaluation of the Shakeout scheme in training deep neural networks on real-life datasets. We first demonstrate that the Shakeout training leads to sparse models as our theoretical analysis implies. Then we show that the sparse models have desirable generalization properties. All the experiments are implemented based on the modifications of Caffe library (Jia et al. 2014).

Shakeout Training and Weight Sparsity

Since Shakeout training implicitly imposes L_1 penalty on the weights, we expect the weights of neural networks learned by Shakeout contain more zeros than those learned by standard back-propagation(BP) (Williams and Hinton 1986) and Dropout (Hinton et al. 2012). This experiment verifies the effect of sparsity by examining an autoencoder model (Baldi 2012) trained on the hand-written image dataset MNIST (LeCun et al. 1998). The autoencoder adopted contains one hidden layer of 256 units, each of which is connected to the 28×28 image pixels and followed by a hyperbolic tangent (i.e. $\tanh$) activation function. The weights talked in this section all refer to those of hidden layers.

To verify the regularization effect, we compare four autoencoders trained under different settings which correspond to standard BP, Dropout($\tau = 0.5$), and Shakeout ($\tau = 0.5$, $c = \{1, 10\}$). All the training methods aim to produce hidden units which can capture good visual features of the handwritten digits. The statistical traits of these different resulting weights are shown in Fig. 2. Moreover, Fig. 3 shows the features captured by each hidden unit of the autoencoders.

As shown in the Fig. 2, the probability density of weights around the zero obtained by standard BP training is quite small compared to the one obtained either by Dropout training or Shakeout training. This indicates the strong regularization effect induced by Dropout and Shakeout training. Furthermore, the sparsity level of weights obtained from training by Shakeout is much higher than the one obtained from training by the Dropout. And under the same τ, increasing c makes the weights much sparser, which is consistent with the characteristic of L_1 penalty induced by Shakeout training (Eq. (4)). Moreover, we can find that due to the induced L_2 regularization, the distribution of weights obtained from training by the Dropout is like a Gaussian, while the one obtained from training by Shakeout is more like a Laplacian because of the additional induced L_1 regularization. Fig. 3 shows that features captured by the hidden units via standard BP training are not directly interpretable, corresponding to insignificant variants in the training data. Both Dropout and Shakeout suppressed irrelevant weights by their regularization effects, where Shakeout produces much sparser and more global features thanks to the combination of L_1 penalty and L_2 penalty.

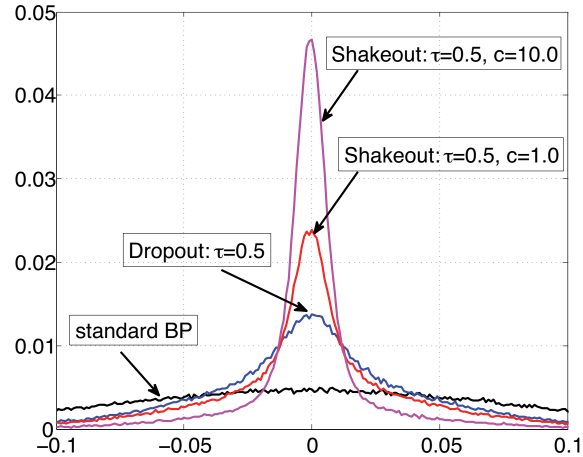

Figure 2: Distributions of the weights of the autoencoder models learned by different training methods. Each curve in the figure shows the frequencies of the weights of an autoencoder taking particular values, i.e. the empirical population densities of the weights. The four curves correspond to four autoencoders learned by standard back-propagation, Dropout ($\tau = 0.5$) and Shakeout ($\tau = 0.5$, $c = \{1, 10\}$). The sparsity of the weights obtained via Shakeout can be seen by comparing the curves.

Classification Experiments

Sparse models often indicate lower complexity and better generalization (Tibshirani 1996; Zou and Hastie 2005; Olshausen and Field 1997). To verify the induced L_1 regularization and subsequent effect of sparse models of the proposed Shakeout scheme, we have applied Shakeout, along with Dropout and standard BP, on training deep neural networks for classification tasks. The tests are performed on two real-life image datasets: the hand-written image dataset MNIST that are used above and the CIFAR-10 image dataset (Krizhevsky and Hinton 2009). MNIST consists of 60k+10k (training+testing) 28×28 images of hand-written digits. CIFAR-10 contains 50k+10k (training +testing) 32×32 images of 10 object classes. In all of our classification experiments, the important hyper-parameters in each training scheme, including (i.e. c and τ in Shakeout, τ in Dropout) are determined by validation. After validation, we train 5 independent networks for each optimal hyper-parameter setting. Once the 5 networks are trained, we report the mean and standard deviation of the classification errors produced by the 5 independent networks. To show the effect against over-fitting, we perform experiments on training sets with different sizes. These training sets are constructed by randomly choosing examples from the original training dataset.

MNIST We have trained two different neural networks, a shallow fully connected one and a deep convolutional one. For the fully-connected neural network, a big hidden layer size is adopted with its value at 4096. The non-linear activation unit adopted is the rectifier linear unit (ReLU). The deep convolutional neural network employed contains two convolutional layers and two fully connected layers. The detailed architecture information of this convolutional neural network is described in Tab. 1.

We separate 10,000 training samples from original training dataset for validation. The results are shown in Tab. 2 and Tab. 3. We can see from the results that when the training data is not sufficient enough, due to over-fitting, all the models perform worse. However, the model trained by Dropout and Shakeout consistently perform better than the one trained by standard BP. Moreover, when the training data is scarce, Shakeout leads to superior model performance compared to the Dropout. Fig. 4 shows the results in

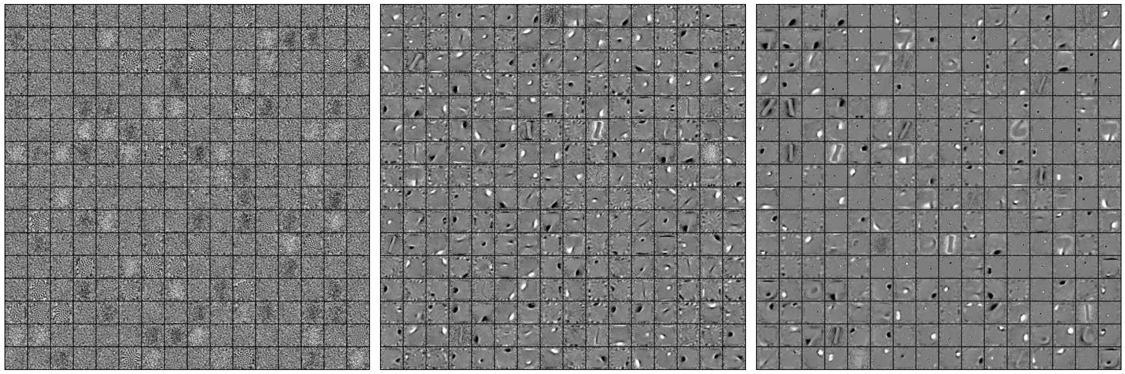

Figure 3: Features captured by the hidden units of the autoencoder models learned by different training methods. The features captured by a hidden unit are represented by a group of weights that connect the image pixels with this corresponding hidden unit. One image patch in a sub-graph corresponds to the features captured by one hidden unit. **Left:** trained by standard BP. **Middle:** trained by Dropout with $\tau = 0.5$. **Right:** trained by Shakeout with $\tau = 0.5$, $c = 0.5$.

Layer	1	2	3	4
Type	conv.	conv.	FC	FC
Channels	20	50	500	10
Filter size	5×5	5×5	-	-
Conv. stride	1	1	-	-
Pooling type	max	max	-	-
Pooling size	2×2	2×2	-	-
Pooling stride	2	2	-	-
Non-linear	ReLU	ReLU	ReLU	Softmax

Table 1: The architecture of convolutional neural network adopted for MNIST classification experiment. Layer 1 and layer 2 are convolutional layers followed by ReLU non-linear activation and max-pooling. Layer 3 and layer 4 are fully-connected layers.

a more intuitive way.

Size	std-BP	Dropout	Shakeout
500	13.66±0.66	11.76±0.09	**10.81**±0.32
1000	8.49±0.23	8.05±0.05	**7.19**±0.15
3000	5.54±0.09	4.87±0.06	**4.60**±0.07
8000	3.57±0.14	**2.95**±0.05	2.96±0.09
20000	2.28±0.09	**1.82**±0.07	1.92±0.06
50000	1.55±0.03	1.36±0.03	**1.35**±0.07

Table 2: Classification on MNIST using training sets of different sizes: fully-connected neural network. Dropout and Shakeout are applied on the hidden units. The table compares the errors of the networks trained by standard back-propagation, Dropout and Shakeout. The mean and standard deviation of the classification errors are obtained by 5 runs of the experiment and are shown in percentage.

CIFAR-10 We use the simple convolutional network feature extractor described in cuda-convnet (layers-80sec.cfg) (Krizhevsky 2012). A 64 unit fully connected layer is imposed on the top of the 3-layer feature extractor. And we apply Dropout and Shakeout operations on this layer. In this experiment, 10,000 colour images are separated from the training dataset for validation and no data augmentation is utilized except that the per-pixel mean computed over the training set is subtracted from each image. We first train for 100 epochs with an initial learning rate of 0.001 and then another 50 epochs with the learning rate of 0.0001.

Size	std-BP	Dropout	Shakeout
500	9.76±0.26	6.16±0.23	**4.83**±0.11
1000	6.73±0.12	4.01±0.16	**3.43**±0.06
3000	2.93±0.10	2.06±0.06	**1.86**±0.13
8000	1.70±0.03	**1.23**±0.13	1.31±0.06
20000	0.97±0.01	0.83±0.06	**0.77**±0.001
50000	0.78±0.05	0.62±0.04	**0.58**±0.10

Table 3: Classification on MNIST using training sets of different sizes: convolutional neural network. Dropout and Shakeout are applied on the hidden units of the fully-connected layer. The table compares the errors of the networks trained by standard back-propagation, Dropout and Shakeout. The mean and standard deviation of the classification errors are obtained by 5 runs of the experiment and are shown in percentage.

As shown in Tab. 4, the performance of models trained by Dropout and Shakeout is consistently superior than the one trained by standard BP. Furthermore, the model trained by Shakeout also outperforms the one trained by Dropout when the training data is scarce. Fig. 5 show the results in a more intuitive way.

Conclusions

We have proposed Shakeout, which is a new regularized training scheme for deep neural networks. We theoretically prove that this training scheme is equivalent to introducing the Elastic Net-like regularization on the weights of the

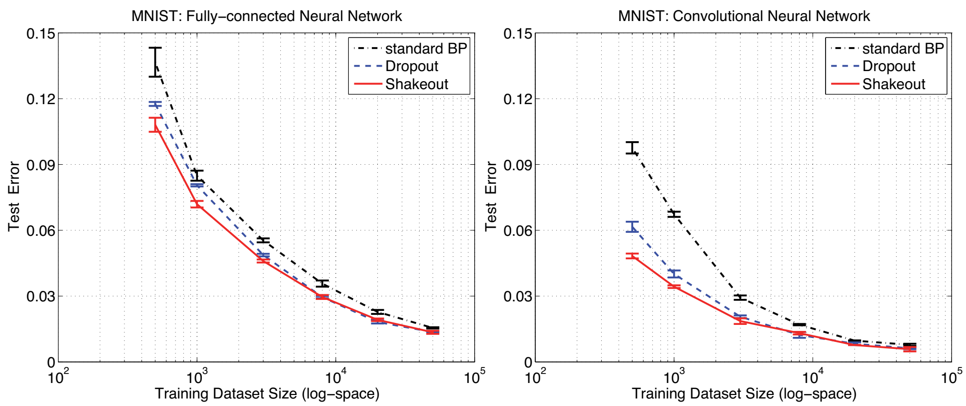

Figure 4: Classification of two kinds of neural networks on MNIST using training sets of different sizes. The curves show the performances of the models trained by standard BP, and those by Dropout and Shakeout applied on the hidden units of the fully-connected layer. **Left:** fully-connected neural network. **Right:** convolutional neural network.

Size	std-BP	Dropout	Shakeout
300	68.26±0.57	65.34±0.75	**63.71**±0.28
700	59.78±0.24	56.04±0.22	**54.66**±0.22
2000	50.73±0.29	46.24±0.49	**44.39**±0.41
5500	41.41±0.52	36.01±0.13	**34.54**±0.31
15000	32.53±0.25	27.28±0.26	**26.53**±0.17
40000	24.48±0.23	**20.50**±0.32	20.56±0.12

Table 4: Classification on CIFAR-10 using training sets of different sizes. Dropout and Shakeout are applied on the hidden units of the fully-connected layer. The table compares the errors of the networks trained by standard back-propagation, Dropout and Shakeout. The mean and deviation of the classification errors are obtained by 5 runs of the experiment and are shown in percentage.

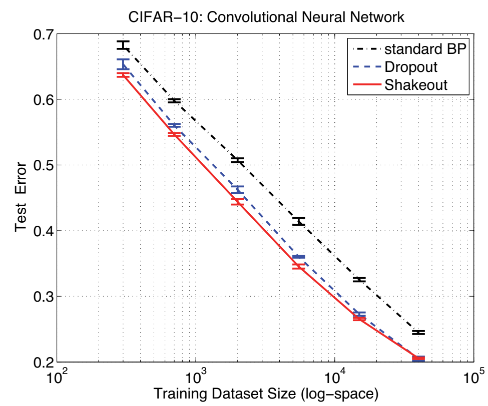

Figure 5: Classification on CIFAR-10 using training sets of different sizes. The curves show the performances of the models trained by standard BP, and those by Dropout and Shakeout applied on the hidden units of the fully-connected layer.

model, which contains a combination of L_1 regularization and L_2 regularization. It is expected that the weights obtained from training by Shakeout is much sparser than the ones obtained by standard back-propagation training and Dropout training, and thus beneficial to the generalization performance of neural network. We experimentally demonstrate the sparsity of the network weights induced by Shakeout. The classification experiments on MNIST and CIFAR-10 show that Shakeout and Dropout help deal with overfitting, where Shakeout outperforms Dropout.

Acknowledgments

This research is supported by Australian Research Council Projects DP-140102164 and FT-130101457.

References

Baldi, P. 2012. Autoencoders, unsupervised learning, and deep architectures. *Unsupervised and Transfer Learning Challenges in Machine Learning, Volume 7* 43.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35(8):1798–1828.

Bengio, Y. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* 2(1):1–127.

Bishop, C. M. 1995. Training with noise is equivalent to tikhonov regularization. *Neural computation* 7(1):108–116.

Chen, N.; Zhu, J.; Chen, J.; and Zhang, B. 2014. Dropout training for support vector machines. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P.; and Bengio, S. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research* 11:625–660.

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

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

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, 675–678. ACM.

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

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Krizhevsky, A. 2012. cuda-convnet. https://code.google.com/p/cuda-convnet/.

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

Lin, M.; Chen, Q.; and Yan, S. 2013. Network in network. *arXiv preprint arXiv:1312.4400*.

Olshausen, B. A., and Field, D. J. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research* 37(23):3311–3325.

Rifai, S.; Glorot, X.; Bengio, Y.; and Vincent, P. 2011. Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*.

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

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2014. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 267–288.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 1096–1103. ACM.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research* 11:3371–3408.

Wager, S.; Wang, S.; and Liang, P. S. 2013. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, 351–359.

Warde-Farley, D.; Goodfellow, I. J.; Courville, A.; and Bengio, Y. 2013. An empirical analysis of dropout in piecewise linear networks. *arXiv preprint arXiv:1312.6197*.

Williams, D. R. G. H. R., and Hinton, G. 1986. Learning representations by back-propagating errors. *Nature* 323–533.

Zhou, T.; Tao, D.; and Wu, X. 2011. Manifold elastic net: a unified framework for sparse dimension reduction. *Data Mining and Knowledge Discovery* 22(3):340–371.

Zou, H., and Hastie, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67(2):301–320.