

Sparseout: Controlling Sparsity in Deep Networks

Najeeb Khan and Ian Stavness

Dept. of Computer Science, University of Saskatchewan, Canada
najeeb.khan@usask.ca
ian.stavness@usask.ca

Abstract. Dropout is commonly used to help reduce overfitting in deep neural networks. Sparsity is a potentially important property of neural networks, but is not explicitly controlled by Dropout-based regularization. In this work, we propose Sparseout a simple and efficient variant of Dropout that can be used to control the sparsity of the activations in a neural network. We theoretically prove that Sparseout is equivalent to an L_q penalty on the features of a generalized linear model and that Dropout is a special case of Sparseout for neural networks. We empirically demonstrate that Sparseout is computationally inexpensive and is able to control the desired level of sparsity in the activations. We evaluated Sparseout on image classification and language modelling tasks to see the effect of sparsity on these tasks. We found that sparsity of the activations is favorable for language modelling performance while image classification benefits from denser activations. Sparseout provides a way to investigate sparsity in state-of-the-art deep learning models. Source code for Sparseout could be found at <https://github.com/najeebkhan/sparseout>.

1 Introduction

Sparsity is often thought to be a desirable property for artificial neural networks. This is likely rooted in early neuroscience studies that discovered sparse coding in the visual cortex [1] hypothesizing that at any given time, only a small number of neurons are used to encode sensory information. Sparsity has been observed both in connectivity [2] and representation [1]. To mimic sparse coding from brain studies, researchers have devised approaches to encourage sparsity when training ANNs.

Sparsity has been used to regularize models by imposing a *sparsity* constraint on the activations of the neural network [3]. Many useful properties are ascribed to sparsity in the literature. It has been hypothesized that neurons that are rarely active are more interpretable than those that are active most of the time [4]; images of natural world objects can be described in terms of sparse statistically independent events; neural networks with sparsity constraints learn filters that resemble the mammalian visual cortex area V1 [1] and area V2 [5]; and sparsity allows faster learning [6].

One of the main motivations behind sparsity based training methods is the biological plausibility of these algorithms. However, recent studies have questioned the pervasiveness of neural sparsity. The biological studies that provide evidence for sparsity are performed when the subject is passive and in reality sparsity might not be the mechanism that the brain uses in active tasks [7]. Empirically for DNNs, new methods that may discourage sparsity, such as Maxout [8] and DARC1 [9], have achieved better performance than sparse methods in certain domains such as computer vision. Therefore it is not clear whether or not sparsity is a generally desirable property for DNNs. We hypothesize that sparsity will benefit some learning tasks and hinder others. Therefore, new DNN training approaches that include the flexibility to either encourage sparsity, where necessary, and discourage sparsity otherwise could provide improved task performance.

There are many approaches for affecting the sparsity of a DNN during training including the use of certain activation functions such as rectified linear units [10]. One of the main ways is through regularization and several deterministic regularization algorithms have been proposed to train deep neural networks with sparse weights [11,12,13] and sparse activations [14,15,16,10,17].

Training deep neural networks with deterministic regularization and back-propagation results in correlated activities of the neurons. To prevent such co-adaptations as well as regularize the models, stochastic regularization methods are used. Stochastic methods such as Dropout, Bridgeout and Shakeout have been shown to be equivalent to ridge, bridge and elastic-net penalties on the model weights. Previous stochastic regularization methods that explicitly encourage sparsity, i.e., Shakeout and Bridgeout require a new set of masked weights per training example in a mini-batch making them computationally expensive. Therefore, these existing methods cannot be applied to large fully connected architectures. Likewise, Shakeout and Bridgeout cannot be easily applied to other convolutional architectures such as DenseNet and Wide-ResNet that provide current state-of-the-art performance for image classification, because they cannot be used with highly optimized black-box implementations such as cuDNN [18].

In this paper, we propose Sparseout, a stochastic regularization method that is capable of either encouraging or discouraging sparsity in deep neural networks. It provides an L_q -norm penalty on the network's activations and therefore can vary activation sparsity by its q parameter. The computational cost of Sparseout is comparable to Dropout and it can be applied to existing optimized CNN and LSTM blocks, making it applicable to state-of-the-art architectures. We provide theoretical and empirical results demonstrating the bridge-regularization capability of Sparseout. We use Sparseout to evaluate whether or not sparsity is beneficial for two distinct learning tasks: image classification and language modeling.

2 Related work

Due to the over-parameterization of deep neural networks, they suffer from large generalization error, specifically, when the dataset size is relatively small. This phenomenon is known as over-fitting. Generalization error is upper bounded by the model complexity [19] thus overfitting could be reduced by controlling the complexity of the model.

One way to control the complexity of a model is to impose constraints on the parameters of the model such as the weights in the neural networks. Such model regularization methods can be classified into deterministic and stochastic methods. Deterministic methods either remove redundant weights [13] or penalize large magnitude weights. Weight penalties are imposed by adding a regularization term to the loss function consisting of a norm of the weight matrix [20].

Stochastic methods randomly perturb the weights so as to achieve minimal co-dependency between neurons [21,22,23] as well as regularizing the model at the same time. Stochastic regularization has become the standard practice in training deep learning models and have outperformed deterministic regularization methods on many tasks. Stochastic regularization techniques have a Bayesian model averaging interpretation as well as they possess an equivalence to weight penalties for linear models. In terms of Bayesian estimation, weight penalties are equivalent to imposing a prior distribution on the model weights.

Beside the weight penalty interpretation, a reason for the effectiveness of stochastic regularization methods could be the prevention of correlated activations. It has been shown that high correlation between activations of the neurons results in overfitting. DeCov [24], reduces overfitting by adding a penalty term to the cost function consisting of the co-variances among the activations of the neurons over a mini-batch.

Another approach to control model complexity, inspired by sparse coding [1], is to impose a *sparsity* constraint on the activations of the neural network [3]. To encourage sparsity of the activations, an L_1 norm of the activations is added to the cost function [5]. Another form of penalty is to add the KL-divergence of the expected activations and a preset target sparsity value ρ [4]. Liao et al. have used a clustering approach to obtain sparse representation by encouraging activations to form clusters [17].

Another related technique that normalizes activations in the network so as to have zero mean and unit variance is Batchnorm [25]. Although, the primary purpose of Batchnorm is accelerating training/optimization of the neural network rather than regularization, Batchnorm has reduced the need for stochastic regularization in certain domains. The above mentioned sparsity-inducing methods are deterministic and thus may result in correlated activations. In this paper we propose Sparseout that implicitly imposes an L_q penalty on the activations thus allowing us to choose the level of sparsity in the activations as well as the stochasticity preventing correlated neural activities. Sparseout is different than Bridgeout [23] in that it is applied to activations rather than the weights. Therefore, Sparseout is orders of magnitude faster and practical than Bridgeout. We believe that Sparseout is the first theoretically-motivated technique that is capa-

ble of simultaneously controlling sparsity in activations and reducing correlations between them, besides being equivalent to Dropout for $q = 2$.

3 Sparseout

Consider a feedforward neural network layer l , the output of l -th is given by

$$\mathbf{a}^l = \sigma(\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l), \quad (1)$$

where \mathbf{W}^l and \mathbf{b}^l are the weight matrix and bias vector for the l -th layer, σ is a non-linear activation function and \mathbf{a}^{l-1} is the output of the previous layer.

The Sparseout perturbed output of the l -th layer $\tilde{\mathbf{a}}^l$ is given by

$$\tilde{\mathbf{a}}_i^l = \begin{cases} \mathbf{a}_i^l - |\mathbf{a}_i^l|^{q/2} & \mathbf{r}_i = 0 \\ \mathbf{a}_i^l + |\mathbf{a}_i^l|^{q/2} \left(\frac{1-p}{p}\right) & \mathbf{r}_i = \frac{1}{p} \end{cases} \quad (2)$$

where \mathbf{r} is a random mask vector randomly sampled from a Bernoulli distribution with probability p and scaled by $1/p$ and q specifies the normed space to which the activations are restricted. Since the random mask is scaled by $1/p$ during training, no changes to the neural network are needed during testing. Since the training of the neural networks is performed using the back-propagated gradients of the error, the gradient of the Sparseout perturbed output is given by

$$\frac{\partial \tilde{\mathbf{a}}_i^l}{\partial \mathbf{a}_i^l} = 1 + \frac{q}{2} |\mathbf{a}_i^l|^{\frac{q}{2}-1} (\mathbf{r}_i - 1) \text{sgn}(\mathbf{a}_i^l), \quad (3)$$

where sgn is the sign function.

Since Sparseout operates on the activations of the neural networks similar to Dropout, Sparseout can be implemented with minimal changes to the existing Dropout implementation. Sparseout can be used with the highly optimized black-box implementations such as cuDNN [18]. The above Sparseout formulation is applicable to any feedforward network layer such as convolutional or fully connected layers as well as layers in recurrent neural networks.

Theorem 1. *Sparseout is equivalent to an L_q penalty on the features of a generalized linear model.*

For a generalized linear model with parameters $\boldsymbol{\beta}$, log partition function A and the perturbed design matrix $\tilde{\mathbf{X}}$ of dimension $n \times d$, the negative log likelihood function could be split into a mean squared error term and a penalty term [26, eq. 6] given by

$$R(\boldsymbol{\beta}) \approx \sum_{i=1}^n \frac{A''(\mathbf{X}_i \cdot \boldsymbol{\beta})}{2} \text{Var}[\tilde{\mathbf{X}}_i \cdot \boldsymbol{\beta}] \quad (4)$$

For the Sparseout perturbation $\tilde{\mathbf{X}}_{i,j} = \mathbf{X}_{i,j} [1 + |\mathbf{X}_{i,j}|^{\frac{q-2}{2}} (\mathbf{r}_j - 1)]$, the variance of $\tilde{\mathbf{X}}_i \cdot \boldsymbol{\beta}$ is given by

$$\text{Var}[\tilde{\mathbf{X}}_i \cdot \boldsymbol{\beta}] = \sum_{j=1}^d \frac{1-p}{p} |\mathbf{X}_{i,j}|^q \beta_j^2 \quad (5)$$

Substituting in Equation 4 we have

$$\hat{R}(\boldsymbol{\beta}) = \frac{1-p}{2p} \|\Gamma \mathbf{X}\|_q^q, \quad (6)$$

where $\Gamma = [\boldsymbol{\beta}^T V(\boldsymbol{\beta}) \boldsymbol{\beta}]^{\frac{2}{q}}$ and $V(\boldsymbol{\beta}) = \text{Diag}(A)$. ■

Theorem 2. *For non-negative activation functions, Dropout is equivalent to Sparseout when $q = 2$.*

Setting $q = 2$ in Equation 2 and considering the fact that \mathbf{a} is non-negative, we have $\tilde{\mathbf{a}}_i^l = \mathbf{r}_i \mathbf{a}_i^l$, which is identical to the Dropout perturbation [21]. ■

L_q -normed spaces with different values of L_q exhibit different sparsity characteristics. For $q < 2$ the norm space is sparse while for $q > 2$ the norm space is dense [27]. With Sparseout we can select the norm space of the activations by choosing the value of the hyper-parameter q . Thus, Sparseout allows us to control the level of sparsity in the activations of the neural networks.

4 Experimental results

4.1 Sparsity characterization

To verify that Sparseout is capable of controlling sparsity of a neural network’s activations, we train an autoencoder with a hidden layer of 512 rectified linear units on the MNIST dataset. Dropout and Sparseout are applied to the hidden layer activations with $p = 0.5$ and different values of q for Sparseout. We measure sparsity of the hidden layer activations during testing (when no perturbations are applied to the activations). To measure sparsity we use the Hoyer’s measure H [28]:

$$H(\mathbf{x}) = \frac{\sqrt{d} - \frac{|\mathbf{x}|_1}{|\mathbf{x}|_2}}{\sqrt{d} - 1} \quad (7)$$

where \mathbf{x} is a d -dimensional vector, $|\mathbf{x}|_1$ is the L_1 -norm and $|\mathbf{x}|_2$ is the L_2 -norm of \mathbf{x} . A vector consisting of equal non-zero values has $H = 0$ while vectors only having one non-zero element has $H = 1$. Figure 1 shows the Hoyer’s sparsity measure on the test set as the training progresses for different values of q . As the value of q decreases below 2, we see an increase in the sparsity of the activations, whereas for q values greater than 2 the sparsity is reduced. For $q = 2$, Sparseout results in the same sparsity as Dropout. These results confirm our theoretical analysis that Sparseout can be used to control sparsity of the activations in the neural networks.

4.2 Computational cost

Sparseout is computationally efficient and incurs similar training cost as Dropout. We train an autoencoder with two hidden layer sizes on MNIST with a batch size of 128 both on Nvidia GTX 1080 GPU. As shown in Table 1, Sparseout is

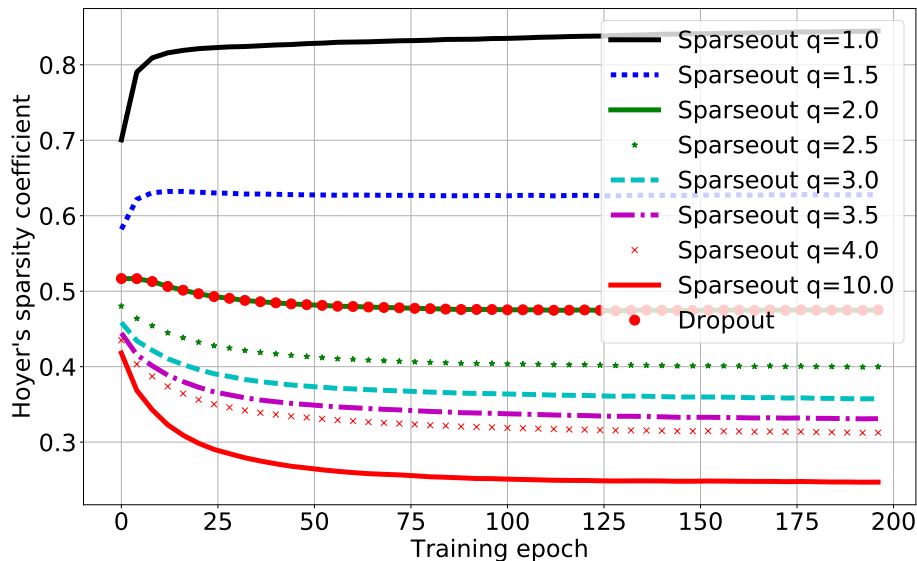


Fig. 1. Average Hoyer’s sparsity measure of the hidden layer activations calculated over the MNIST test set for an autoencoder trained on MNIST with Dropout versus Sparseout with different values of q .

only fractionally more expensive than Dropout while Bridgeout is an order of magnitude more expensive even for this simple model. Also doubling the hidden layer size results in a doubling of the execution time for Bridgeout while Sparseout and Dropout have almost constant execution time due to utilization of GPU parallelism.

Table 1. Average execution time in seconds per epoch for different types of stochastic regularization for an autoencoder with different hidden layer sizes.

Hidden layer size	Backprop	Dropout	Sparseout	Bridgeout
1024	5.2	5.3	5.8	31.6
2048	5.6	5.6	6.0	57.2

4.3 Image classification

Image classification is one of the key areas where deep neural networks have been highly successful achieving state-of-the-art results. The CIFAR datasets [29] are a standard benchmark for image classification. The CIFAR-10 dataset consists of color images of size 32×32 each belonging to one of the ten classes of objects. The dataset is divided into a training set of 50000 images and a test

set of 10000 images. The CIFAR-100 dataset is similar to CIFAR-10 except that the images are divided into 100 classes of objects, thus making the classification task more harder than CIFAR-10. We used the standard pre-processing of mean and standard deviation normalization. Random cropping and random horizontal flips were used for data augmentation.

We use the wide residual network (WRN) architecture to evaluate the effect of sparsity on classification accuracy using Sparseout. WRNs achieved state-of-the-art accuracy on several image classification tasks including CIFAR-10 and CIFAR-100. WRNs are based on deep residual networks [30] that use identity links between the input and output of each layer known as the residual connections, but they employ fewer and wider layers. The residual connections helps in training very deep neural networks consisting of upto a thousand layers.

We employ a WRN with the basic building block shown in Figure 2. The stochastic regularization is applied between the convolutional layers. Each convolutional layer is preceded by batch normalization and rectified linear unit activation function. In our experiments we use the WRN architecture WRN-28-10 with depth 28 and a widening factor of 10. A Dropout probability of $p = 0.3$ was used. Stochastic gradient descent with a mini-batch of 64 was used to train the networks. The learning rate was annealed from 0.1 at 60, 120 and 160 epochs by a factor of 0.2 as in the original WRN paper [31].

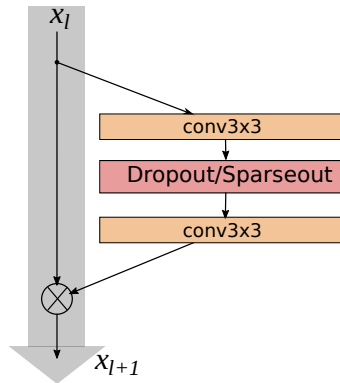


Fig. 2. The basic building block of wide residual network architecture with either Dropout or Sparseout stochastic regularization. Figure adapted from [31]

Image Classification Results For image classification we found that Sparseout with $q > 2$ resulted in better performance compared to values of $q < 2$ as shown in Figure 3. For $q < 2$ the accuracy drops as the training progresses beyond around 100 epochs indicating overfitting. As shown in Table 2, error rate for $q = 2.5$ is about 1 percent lower than Dropout for CIFAR-10 and 2.5 per-

cent lower for CIFAR-100. Our baseline results are comparable to the baselines reported in the literature for CIFAR-100 and better for CIFAR-10 [31,32].

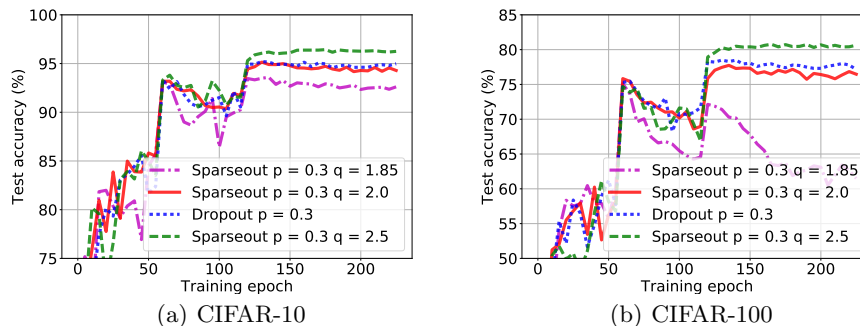


Fig. 3. Test accuracy during training of WRN on CIFAR-10 and CIFAR-100 for Dropout versus Sparseout with different values of q .

Table 2. Mean test errors for the same WRN network trained with Dropout and Sparseout with different values of q . Lower sparsity (high q) results in lower test error.

Model	p	q	CIFAR10	CIFAR100
WRN-28-10-Sparseout	0.3	1.5	7.58	25.87
WRN-28-10-Sparseout	0.3	1.85	5.92	24.65
WRN-28-10-Sparseout	0.3	2.0	4.72	21.91
WRN-28-10-Dropout	0.3	-	4.59	21.66
WRN-28-10-Sparseout	0.3	2.5	3.63	19.07
WRN-28-10-Dropout [31]	$p=0.3$	-	3.89	18.85
WRN-28-10- $L_{0_{hc}}$ [32]	-	-	3.83	18.75

4.4 Language Modelling

Another task for which deep learning has been widely used is natural language processing (NLP). The dimensionality of NLP tasks is very high and sparse; therefore, sparsity is likely to play an important role in such tasks. Language modelling (LM) assigns a probability to a sequence of words. LM is an important component of several NLP tasks such as speech recognition, information retrieval and machine translation among others. Since LM is a sequential task recurrent neural networks are used for it. Vanilla RNN are difficult to train due to vanishing and exploding gradients problem. To overcome these limitations, long short term memory (LSTM) models are used instead [33].

The LSTM model is a type of recurrent neural network with layers consisting of *memory cells*. The weights of the nodes in a memory cell learn the long term information while a node with a self-connected edge retains short term information. The input gate, forgetting gate and output gate help in controlling the flow of information in the LSTM. For a detailed review of the LSTM formulation see Lipton et al. [34].

We adapt the baseline LSTM architecture for word level language modelling from Merity et al. [35]. The model consists of 3 layers of 1150 units. To train the baseline model we used the same hyper-parameters used by Merity et al.¹ except that we used only stochastic gradient descent for training.

We replace variants of Dropout with variants of Sparseout in the LSTM model. Variational Dropout [36] is replaced with variational Sparseout where a single random mask is used within a forward and backward pass. Embedding Dropout applied to the word embedding layer is similarly replaced with embedding Sparseout.

We evaluate the model on two standard word-level language modelling datasets where the task is to predict the next word and the performance is evaluated on perplexity which is the negative log likelihood raised to the exponent. The first dataset is the Penn Treebank dataset [37] that contains 1 million words and a vocabulary size of 10,000. The second dataset is the WikiText-2 dataset [38] which contains over 100 million words and a vocabulary of size 30,000.

Language modeling results Applying Sparseout with $q > 2$ resulted in significant overfitting as shown in Figure 4. For $q < 2$ we found that Sparseout resulted in better prediction performance than Dropout. For PTB dataset Sparseout results in 2.5 percent reduction in relative perplexity. For Wiki-2 dataset the reduction in relative perplexity is 1.25 percent as shown in Table 3.

Table 3. Single model test perplexities on PTB and Wiki-2 datasets for LSTM models trained with Dropout, Sparseout with $q = 2.25$ to reduce sparsity, and Sparseout with $q = 1.75$ to increase sparsity. Higher sparsity results in lower (better) relative perplexity.

Model	Penn Tree Bank	WikiText-2
LSTM-Sparseout ($q = 2.25$)	62.7	70.18
LSTM-Dropout	62.13	68.34
LSTM-Sparseout ($q = 1.75$)	60.57	67.17
AWD-LSTM-Dropout [35]	57.3	65.8

5 Discussion

Existing literature is contradictory on whether sparsity is a good [14,15,16,10,17,32] or bad [7,39,9,8,40] property for deep neural networks. No previous study has

¹ <https://github.com/salesforce/awd-lstm-lm>

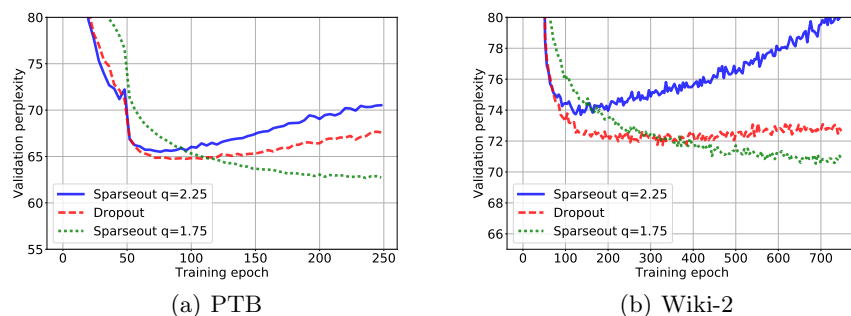


Fig. 4. Validation perplexity for language modeling on PTB and Wiki-2 datasets for LSTM models trained with Dropout, Sparseout with $q = 2.25$ to reduce sparsity, and Sparseout with $q = 1.75$ to increase sparsity.

evaluated sparse vs. non-sparse networks in a controlled fashion with stochastic regularization. In this study, we propose a new bridge-regularization scheme, Sparseout, which has the flexibility to control sparsity and the efficiency to be applied to large networks.

We evaluated Sparseout with two distinct network architectures and machine learning tasks: CNNs for image classification and LSTMs for language modeling. Our empirical results show that lower sparsity improves image classification performance, whereas higher sparsity improves performance on language modeling. These results align with the fundamental differences between data types: relatively tiny densely-featured images vs. sparsely-featured high-dimensional language data.

In this study, we chose the most suitable architecture for each task: CNNs for IID image classification and RNNs for sequential language modelling. Therefore, we evaluated task-architecture in a coupled manner. For each task, image classification or language modelling, we tested two datasets (CIFAR10/CIFAR100 or PTB/WikiText-2) and obtained consistent results regarding the benefit or lack thereof of sparse activations. It is possible, however, that the inherent sparse nature of convolutional layers requires spreading of the activations over all the neurons while enforced parsimony of representation is helpful for the fully connected gates in an LSTM. Therefore, decoupling the effect of data type from that of architecture is an important consideration we plan to investigate as future work.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research* **37**(23) (1997) 3311–3325
2. Morris, G., Nevet, A., Bergman, H.: Anatomical funneling, sparse connectivity and redundancy reduction in the neural networks of the basal ganglia. *Journal of Physiology-Paris* **97**(4) (2003) 581–589
3. Thom, M., Palm, G.: Sparse activity and sparse connectivity in supervised learning. *Journal of Machine Learning Research* **14**(Apr) (2013) 1091–1143
4. Hinton, G.: A practical guide to training restricted Boltzmann machines. *Momentum* **9**(1) (2010) 926
5. Lee, H., Ekanadham, C., Ng, A.Y.: Sparse deep belief net model for visual area V2. In: *Advances in neural information processing systems*. (2008) 873–880
6. Schweighofer, N., Doya, K., Lay, F.: Unsupervised learning of granule cell sparse codes enhances cerebellar adaptive control. *Neuroscience* **103**(1) (2001) 35–50
7. Spanne, A., Jörntell, H.: Questioning the role of sparse coding in the brain. *Trends in neurosciences* **38**(7) (2015) 417–427
8. Goodfellow, I., Warde-farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. (2013) 1319–1327
9. Kawaguchi, K., Kaelbling, L.P., Bengio, Y.: Generalization in deep learning. *arXiv preprint arXiv:1710.05468* (2017)
10. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. (2011) 315–323
11. Hanson, S.J., Pratt, L.Y.: Comparing biases for minimal network construction with back-propagation. In: *Advances in neural information processing systems*. (1989) 177–185
12. LeCun, Y., Denker, J.S., Solla, S.A., Howard, R.E., Jackel, L.D.: Optimal brain damage. In: *NIPS*. Volume 2. (1989) 598–605
13. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: *Advances in Neural Information Processing Systems*. (2015) 1135–1143
14. Chauvin, Y.: A back-propagation algorithm with optimal use of hidden units. In: *Advances in neural information processing systems*. (1989) 519–526
15. Mrázová, I., Wang, D.: Improved generalization of neural classifiers with enforced internal representation. *Neurocomputing* **70**(16) (2007) 2940–2952
16. Wan, W., Mabou, S., Shimada, K., Hirasawa, K., Hu, J.: Enhancing the generalization ability of neural networks through controlling the hidden layers. *Applied Soft Computing* **9**(1) (2009) 404–414
17. Liao, R., Schwing, A., Zemel, R., Urtasun, R.: Learning deep parsimonious representations. In: *Advances in Neural Information Processing Systems*. (2016) 5076–5084
18. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E.: cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759* (2014)
19. Shalev-Shwartz, S., Ben-David, S.: *Understanding machine learning: From theory to algorithms*. Cambridge university press (2014)
20. Neyshabur, B., Tomioka, R., Srebro, N.: Norm-based capacity control in neural networks. In: *Conference on Learning Theory*. (2015) 1376–1401

21. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* **15**(1) (2014) 1929–1958
22. Kang, G., Li, J., Tao, D.: Shakeout: A new regularized deep neural network training scheme. In: *Thirtieth AAAI Conference on Artificial Intelligence*. (2016)
23. Khan, N., Shah, J., Stavness, I.: Bridgeout: stochastic bridge regularization for deep neural networks. *arXiv preprint arXiv:1804.08042* (2018)
24. Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., Batra, D.: Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068* (2015)
25. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning*. (2015) 448–456
26. Wager, S., Wang, S., Liang, P.S.: Dropout training as adaptive regularization. In: *Advances in neural information processing systems*. (2013) 351–359
27. Park, C., Yoon, Y.J.: Bridge regression: adaptivity and group selection. *Journal of Statistical Planning and Inference* **141**(11) (2011) 3506–3519
28. Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research* **5**(Nov) (2004) 1457–1469
29. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, Citeseer (2009)
30. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. (2016) 770–778
31. Zagoruyko, S., Komodakis, N.: Wide residual networks. *arXiv preprint arXiv:1605.07146* (2016)
32. Louizos, C., Welling, M., Kingma, D.P.: Learning sparse neural networks through l_0 regularization. In: *International Conference on Learning Representations*. (2018)
33. Sundermeyer, M., Schlüter, R., Ney, H.: Lstm neural networks for language modeling. In: *Thirteenth Annual Conference of the International Speech Communication Association*. (2012)
34. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* (2015)
35. Merity, S., Keskar, N.S., Socher, R.: Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182* (2017)
36. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Insights and applications. In: *Deep Learning Workshop, ICML*. (2015)
37. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.* **19**(2) (June 1993) 313–330
38. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016)
39. Rigamonti, R., Brown, M.A., Lepetit, V.: Are sparse representations really relevant for image classification? In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE* (2011) 1545–1552
40. Gulcehre, C., Cho, K., Pascanu, R., Bengio, Y.: Learned-norm pooling for deep feedforward and recurrent neural networks. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer* (2014) 530–546