Modified Dropout for Training Neural Network

James Duyck
Machine Learning Department
jduyck@andrew.cmu.edu

Min Hyung Lee
Machine Learning Department
minhyunl@andrew.cmu.edu

Eric Lei
Machine Learning Department
elei@andrew.cmu.edu

Abstract

Dropout is a method that prevents overfitting when training deep neural networks. It involves sampling different sub-networks by temporarily removing nodes at random. Dropout works well in practice, but its properties have not been fully explored or theoretically justified. Our project explores the properties of dropout by applying methods used in optimization such as simulated annealing and low discrepancy sampling in model combination. We include experimental results that demonstrate the properties of these modifications to dropout and compare them to existing versions of dropout.

1 Introduction

Deep neural networks are multi-layer feed-forward neural networks. With multiple layers of neurons, deep neural networks can model complex non-linear relationships between variables without requiring exponentially many nodes per layer. However, due to the complexity of the models involved, there is a danger of overfitting when training deep neural networks [1]. Without the large amount of data needed to train the whole model, small spurious relationships between variables are captured through the neural network. As such, deep neural networks with more than one hidden layer are prone to inaccurate predictions.

Many different methods have been developed to prevent overfitting. The method most used in other domains to prevent overfitting is regularization, and according to Srivastava et al. [1], the best method of regularizing a fixed size model is to combine the result of multiple networks trained on different parameter settings [1]. However, training multiple neural networks and combining the results is computationally expensive and thus infeasible.

The dropout method combines the results of different networks by randomly dropping out neurons in the network. Since it shares parameters across networks via the full network, this method uses less computation in training, preventing overfitting at a low cost. In a regular neural networks, the network parameters are learned through the backpropagation algorithm, in which stochastic gradient descent is applied over mini-batches of the training dataset. In the dropout method, for each mini-batch, a thinned network is sampled from the complete set of possible networks by deleting each perceptron with a probability of p. Gradient descent is then applied on the thinned network. After training, the learned parameters are used for the classification of new data points. This new method of regularization leads to significant improvements in the performance of the algorithm in various applications, although it requires two to three times the original training time [1]. Such improvements led to dropouts success in many applications including image classification and automatic speech recognition [2].

Although dropout performs well in practice and has been shown to reduce overfitting, there are various explanations for how dropout prevents overfitting and why dropout is more effective than other methods for reducing overfitting [1]. The objective of this study is to determine which explanations are more likely to be correct. We do this by applying methods used in other areas of optimization, which we would expect to improve or not improve dropout, depending on whether the corresponding explanation is correct.

In section 2, we detail some prior work done to extend the dropout method. In section 3, we discuss hypotheses for why dropout performs well and modifications that we have made to the dropout algorithm in order to test these hypotheses. In section 4, we report experimental results on the modifications described in section 3. We then discuss the implications of these results towards the hypotheses. In section 5, we specify the future direction of our project, including further theoretical work needed to justify our experimental results and other approaches that might be tried to improve dropout.

2 Related Work

Starting with the ImageNet challenge, deep neural networks with dropout have been proven to perform well in many high-dimensional classification problems. Following its introduction, some work has been done on improving the performance of dropout. One intuitive extension is proposed by Li et al. [3]. Their method, called DropConnect, applies a dropout mask on the outgoing edges of each node, dropping an outgoing edge with probability 0.5. Dropout can be seen as a special case of DropConnect in which all of the outgoing edges from one node are dropped once it is chosen. Through DropConnect, the number of different networks that can be sampled increases significantly. It is reported that DropConnect produces state-of-the-art performance in several challenges.

Ba and Frey [4] make another extension to the dropout method. The proposed adaptive method, called the standout method, probabilistically finds the optimal dropout rate for a given node based on the results of the previous layers. The process is equivalent to creating and learning a separate belief network of dropout rates, on top of the existing neural network. Through the standout method, nodes that hinder performance are given lower dropout rates, leading to a high probability of the node being dropped, and vice versa. As a result, the method performs significantly better than the original dropout method in several challenges.

Our modifications to dropout is based on the paper by Baldi and Sadowski [5]. In the paper, Baldi and Sadowski show that the expectation of the dropout gradient is approximately equal to a regularization of the gradient of the network weighted. They also show that using dropout is a form of model averaging, where the result of averaging the weights trained on different network structures is approximately equivalent to averaging the result of each network structure. Based on these two explanations, we modify dropout using concepts used in regularization and model combination.

3 Methods

3.1 Changing dropout rate

As mentioned above, dropout adds Bernoulli noise to the nodes of a neural network, and it can be shown that this is a stochastic way of regularizing the weights. Regularization in general acts to prevent overfitting by favoring simpler models over the optimum at the training data. A different way that noise can be used to improve performance in optimization is by allowing the algorithm to escape from local minima. This is similar to simulated annealing [6], in which a function is optimized by updating a starting state to a new state with a probability dependent on the difference in function values between the two states. A global temperature is defined, which starts at some high value and decreases to 0 over updates. As the temperature decreases, the probability of accepting a new state with a lower function value or which is farther from the current state decreases. At the beginning of the process, the state makes large scale changes, allowing it to escape from local optima. At the temperature decreases, the state makes small scale changes, moving towards a better optimum.

In this work we apply a similar idea to dropout noise. When there is a high dropout rate (low retention rate), the noise will be higher. This will cause the output of the network to vary more widely, which will cause the stochastic gradient descent backpropagation algorithm to produce more extreme changes. When the dropout rate is lower (high retention rate), the backpropagation algorithm will produce smaller scale changes. By starting with a low retention rate and increasing it, dropout may be able to escape from poor local minima while still converging to a more optimal minimum.

We apply this idea in two ways. In the first method, we increase the retention rate p over epochs. We try increasing p using convex, linear, and concave functions of the epoch. We also try similar functions with decreasing p for comparison. These functions are shown in Figure 1. We refer to these methods as *increasing dropout* or *decreasing dropout*. We increase p from 0.5 to 1.0. In the second method, we assign a bound p' on p, which increases linearly from 0.3 or 0.7 to 1.0. Then, for each mini-batch, p is chosen from a uniform random distribution between p' and 1.0. We refer to this method as *bounded random dropout*.

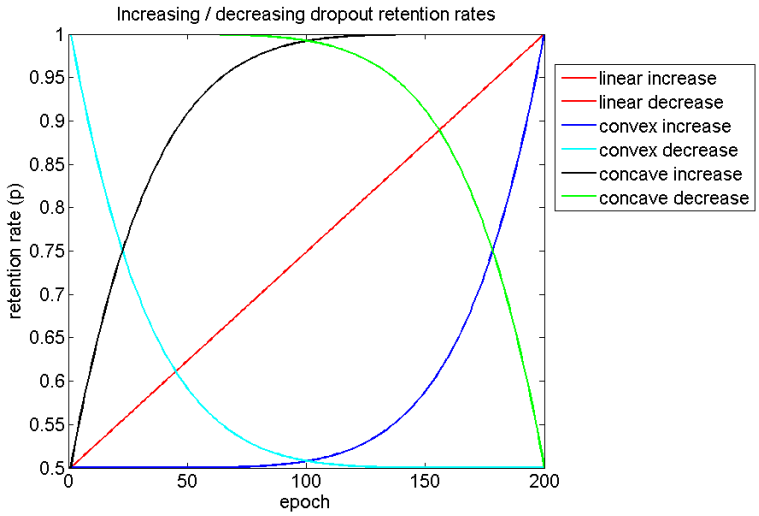


Figure 1: We changed the retention rate p using different functions over epochs.

3.2 Dropout as Model Combination

Another way of viewing dropout is as model combination. The dropout method runs gradient descent using different networks sampled by dropping out nodes, and the weights learned are averaged in the end. Considering each sampled network as a model to be learned, the method combines the learned parameters of different models when obtaining the final result.

According to Brown et al. [7], one of the keys to achieving high performance when using an ensemble of classifiers is to create a diverse set of classifiers. Since the dropout method averages the weight parameters and not the final result of the neural network classifier, dropout is not an ensemble of classifiers. However, we conjecture that promoting diversity in the models would improve the performance of the dropout method. Accordingly, two different methods were created based on the conjectures, which differ in how the diverse set of networks is sampled.

3.2.1 Alternating Dropout

The first method samples the network in each iteration based on the network sampled in the previous iteration, making the current network different from the previously sampled network. The algorithm works as follows. First, a network is sampled using a pre-defined dropout rate, where each node is dropped out with uniform probability. Then, the subsequent networks are sampled by adding a bias towards making a different action for each node. A node that was dropped out in the previous iteration is dropped out with a low probability, and a node that was retained is dropped out with a high probability. Define p_{dd} and p_{rd} as the probability of dropping out a node dropped and retained

in the previous layer respectively. To make the dropout rate constant p over iterations, the following property has to be satisfied

$$\frac{1-p}{p} = \frac{1-p_{dd}}{p_{rd}}$$

Note that if $p_{dd} = p$, there is no difference from normal dropout, so p_{dd} was set to 0 to create the maximum amount of difference from normal dropout. Moreover, through experiments testing different values of p_{dd}, the results were did not differ significantly from regular dropout when p_{dd} was close to p. Thus, the algorithm always retains a node that was dropped out in the previous iteration and drops out a node that was retained in the previous iteration with probability $\frac{p}{1-p}$. We call this process "alternating dropout."

3.2.2 Low-discrepancy method

Another method we devised uses a deterministic approach, where given a number of networks that will be sampled, a set of diverse networks is created. To do this, low-discrepancy sequences are used.

A low-discrepancy sequence is a deterministic sequence that is more equidistributed in a given space compared to pseudo-random sequences. Formally, for a given space, low-discrepancy sequences minimize the difference between the proportion of points in a subspace and the proportion of the volume of the subspace to the volume of a space, for any subspace. Thus, the points are evenly distributed inside the space [8].

A typical application of low-discrepancy sequences is numerical integration, where the integral is approximated as the sum of the integrand evaluated over the points in the low-discrepancy sequence. Plain Monte Carlo, in contrast, would approximate the integral by summing over the points sampled randomly. The advantage of quasi-Monte Carlo is faster convergence: its convergence rate tends to be $O(N^{-1})$ compared to $O(N^{-1/2})$ for Monte Carlo, where N is the number of points in the sequence [9].

A similar concept can be applied to dropout. If a network can be sampled using low-discrepancy sequences, the equally distributed nature of low-discrepancy sequences allow greater diversity in the set of networks sampled. Each network sampled through dropout can be expressed as a binary vector, in which each element of the vector corresponds to each node in the network. For each node, if the corresponding value in the vector is 1, the node is retained, and if it is 0, the node is dropped out. To get a sequence of binary vectors, we use Sobol and Halton sequences. These low-discrepancy sequences sample equally distributed vectors from a unit space. These vectors are then converted to binary vectors by thresholding each element with the pre-defined dropout rate. The sampled vectors are then used to construct networks, which are then used to train the weight parameters.

4 Experiments

4.1 Procedure

We used nine UCI classification datasets for our experiments. In Gisette the task is to distinguish between digits 4 and 9. In letter recognition and ISOLET the task is ti distinguish between letters A-Z. In MAGIC gamma telescope the task is to determine whether photons are generated by gamma rays or cosmic rays. In CNAE-9 the task is to categorize Brazilian companies according to text descriptions. In page blocks the task is to categorize segmented blocks from pages. In wall-following robot the task is to determine the direction a robot should turn. In Statlog shuttle the task is to classify the condition of a space shuttle. In multiple features the task is to distinguish between digits 0-9.

We implemented feed-forward neural networks with dropout, basing our code on an open-source library [10]. Our approach utilized backpropagation. For our experiments, we trained a network with three hidden layers with 64, 16, and 4 nodes respectively. Note that the small number of nodes in the final layer could lead to inaccurate predictions on datasets with many classes, such as letter recognition and ISOLET. Our learning rate—the constant in stochastic gradient descent—was 1,

dataset	n	d	number of classes
gisette	4000	5000	2
letter recognition	15000	16	26
magic	12680	10	2
cnae	720	856	9
pageblocks	3648	10	5
wallrobot	3637	24	4
shuttle	43500	9	7
isolet	6238	617	26
multiple features	1333	240	10

Table 1: Size of datasets.

the mini-batch size 10, and the number of epochs 200. These hyperparameters were chosen using cross-validation on the Gisette training data.

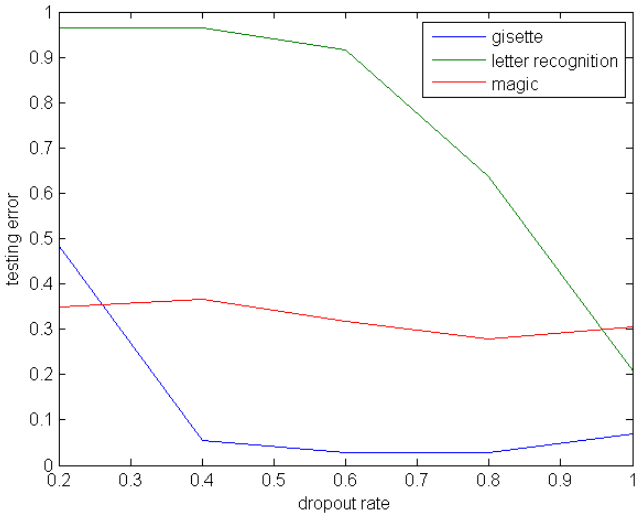


Figure 2: Classification error on three datasets for different retention rates.

Here we compare classification error of a standard neural network (no dropout), normal droput, and three modifications of dropout. For normal dropout, the best error from a range of dropout values is presented (see Figure 2; the best retention rate tends to be near 1. The first modification was to change the retention rate from 0.5 to 1 or 1 to 0.5 over training epochs with linear, convex, or concave functions. The second and third methods were alternating and low-discrepancy dropout.

4.2 Results

On five of nine datasets, dropout outperformed no dropout (see Table 2). The exceptions were letter recognition, ISOLET, CNAE, and wall robot. The first two can be explained by the higher number of classes: 26. Since the final layer only had four nodes, the network probably was not sufficiently expressive for these datasets. However, we currently do not have a satisfactory explanation for why no dropout was significantly better than normal dropout on the CNAE and wall robot datasets.

Overall, increasing or decreasing the retention rate over training iterations does not appear to give good performance (see Table 2). In fact, the error rates are often 15

Alternating outperformed dropout on eight datasets, while low-discrepancy outperformed on seven. The two methods had roughly equivalent performance. The greatest underperformance was on the shuttle dataset, on which normal dropout had 3.9% error, alternating 5.2%, and low-discrepancy

dataset	no dropout	dropout	linear decrease	convex decrease
gisette	6.80%	2.80%	3.05%	2.90%
letter recognition	20.20%	63.70%	78.90%	94.40%
magic	31.30%	27.80%	28.40%	29.70%
cnae	13.60%	32.20%	35.30%	71.90%
pageblocks	9.53%	9.50%	9.53%	7.34%
wallrobot	10.30%	21.60%	30.40%	32.70%
shuttle	15.30%	3.88%	13.50%	15.40%
isolet	10.20%	53.00%	54.90%	80.20%
dataset	concave decrease	linear increase	convex increase	concave increase
gisette	3.65%	2.85%	3.15%	3.60%
letter recognition	72.10%	51.50%	86.70%	22.70%
magic	30.10%	28.60%	30.70%	31.70%
cnae	16.40%	18.90%	61.40%	11.40%
pageblocks	7.29%	9.53%	6.90%	8.66%
wallrobot	38.20%	10.90%	28.60%	8.08%
shuttle	14.60%	9.65%	7.99%	7.74%
isolet	44.00%	55.90%	79.00%	23.70%

Table 2: Classification error across different datasets for different methods for increasing or decreasing the retention rate.

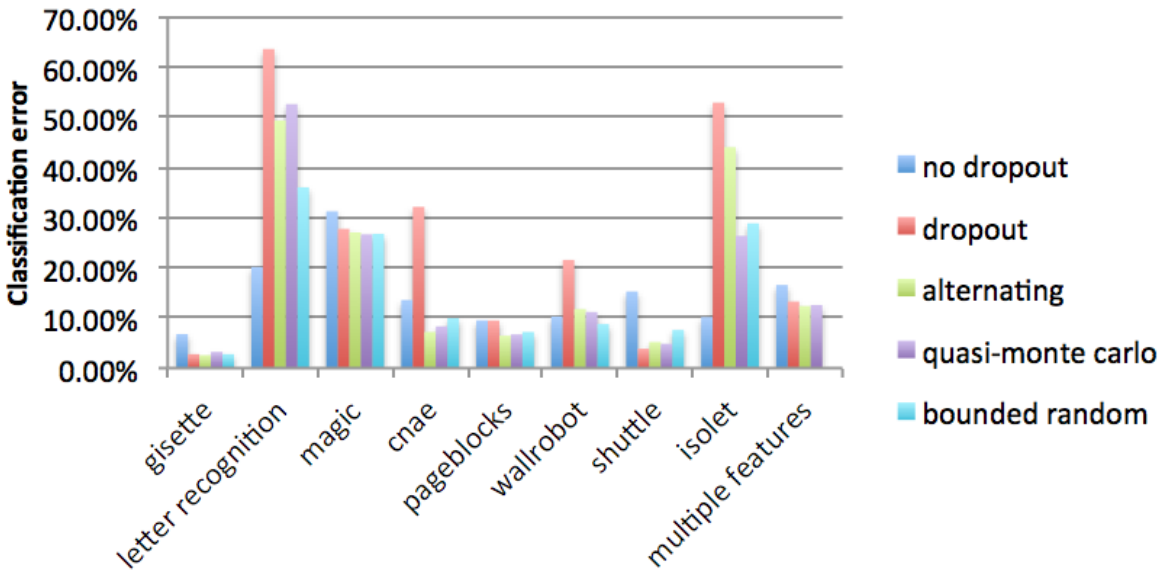


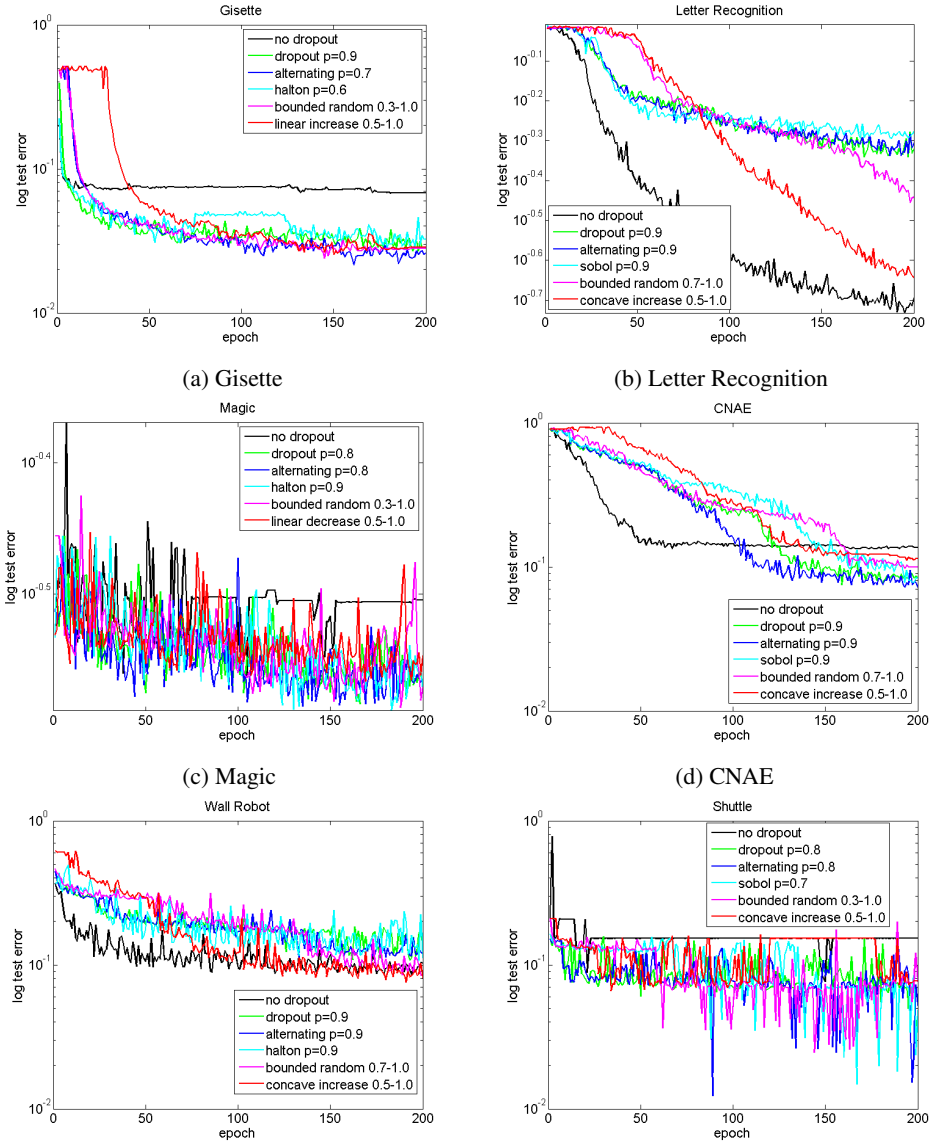
Figure 3: Classification error across different datasets for alternating and low-discrepancy methods.

4.9%. These differences are fairly small and signify that these methods perform equal to or better than normal dropout overall.

We observed that no dropout tended to require fewer training iterations. Figure 4 illustrates how the testing error of no dropout converges more quickly than normal dropout, increasing retention, alternating, and low-discrepancy. This discrepancy can be attributed to the lack of artificial noise added by dropout. Of course, the disadvantage of this faster convergence is greater error. Moreover, the other methods converge at roughly the same pace, demonstrating that the modifications to dropout do not extend training time.

4.3 Discussion

Recall that a concave increasing retention rate had better performance than other ways of changing the retention rate. This may be true for a few reasons. Note that the concave function spends



(a) Gisette

(b) Letter Recognition

(c) Magic

(d) CNAE

(e) Wall Robot

(f) Shuttle

Figure 4: Test error over training iterations for different dropout methods.

a small number of epochs with a low retention rate and more epochs with a high retention rate (Figure 1). When concave increasing performed well, higher retention rates (about 0.8-0.9) also performed better than lower retention rates. Since for concave increasing dropout, the retention rate spends more epochs near the more successful retention rates in normal dropout, we expect concave increasing to be successful. Additionally, it may be the case that when less time is spent with high retention rates, there is not enough time for the algorithm to converge. However, the concave increasing rate did not perform significantly better than normal dropout, only about equally. This result may indicate that local optima are not a large problem.

The relatively greater success of alternating and low-discrepancy methods is interesting when we juxtapose their advantages with the number of training instances. For example, on the three largest training sets normal dropout outperformed both these modifications. Conversely, on multiple features, one of the smallest training sets, normal dropout was outperformed by both modifications. Our interpretation of this result is that a lack of diversity in network structures is a problem in normal dropout. The number of network structures used is proportional to the number of training instances. Thus the network structures for smaller training sets could lack diversity. If we view dropout as a form of model averaging, then we would expect it to perform worse with fewer network structures used. Since the alternating and low-discrepancy methods appear to improve performance on smaller training sets, this result suggests that it is at least somewhat correct to consider dropout as model averaging. Additionally, these modifications may be more practical in situations where training data is limited.

5 Conclusions and Future Work

We explored several proposed hypotheses explaining the success of dropout. These hypotheses served as the basis for three modifications to dropout. Each modification was tested on many datasets, and its performance was compared to that of no dropout and normal dropout. Our experiments suggest that local optima are not a significant problem in dropout. They also indicate that dropout may perform very poorly with less training data. Two of our modifications help fix this problem by promoting diversity in the network structures used in dropout.

In the future, it would be desirable to derive theoretical justifications for why these methods do or do not work. An important problem is to investigate the relationship between diversity of network structures used and the bias and variance of the resulting classifier. To do this, one would need to quantify network diversity and possibly show that the alternating or low-discrepancy methods lead to high diversity. Another possible avenue would be the problem of automatically learning the optimal dropout rate during training. One Bayesian approach would be to treat the optimal rate as a random variable and use some criterion to update one's beliefs after each training iteration.

6 Acknowledgements

This basis for this project idea was proposed by Andrew Wilson.

References

[1] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html

[2] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[3] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, R. Fergus, Regularization of neural networks using dropconnect, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1058–1066.

[4] J. Ba, B. Frey, Adaptive dropout for training deep neural networks, in: Advances in Neural Information Processing Systems, 2013, pp. 3084–3092.

[5] P. Baldi, P. J. Sadowski, Understanding dropout, in: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (Eds.), Advances in Neural Information Processing Systems 26, Curran Associates, Inc., 2013, pp. 2814–2822.
URL http://papers.nips.cc/paper/4878-understanding-dropout.pdf

[6] W. L. Goffe, G. D. Ferrier, J. Rogers, Global optimization of statistical functions with simulated annealing, Journal of Econometrics 60 (1) (1994) 65–99.

[7] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: a survey and categorisation, Information Fusion 6 (1) (2005) 5–20.

[8] I. L. Dalal, D. Stefan, J. Harwayne-Gidansky, Low discrepancy sequences for monte carlo simulations on reconfigurable platforms, in: Application-Specific Systems, Architectures and Processors, 2008. ASAP 2008. International Conference on, IEEE, 2008, pp. 108–113.

[9] S. Asmussen, P. W. Glynn, Stochastic Simulation: Algorithms and Analysis: Algorithms and Analysis, Vol. 57, Springer, 2007.

[10] R. B. Palm, Prediction as a candidate for learning deep hierarchical models of data, 2012.