Determining Optimum Drop-out Rate for Neural Networks

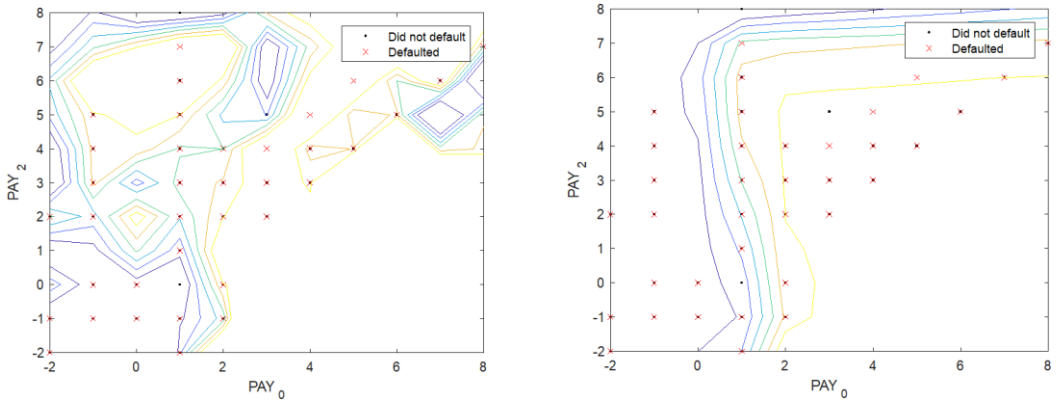
Alexander Pauls
Josiah A Yoder
Department of Electrical Engineering and Computer Science
Milwaukee School of Engineering
1025 North Broadway
Milwaukee WI 53202-3109
paulsaj@msoe.edu

Abstract

Dropout is used to reduce overfitting in neural networks. Past research determines the optimum dropout rate for a dataset but does not compare optimal dropout rates across datasets. The purpose of this project is to investigate a correlation in optimum dropout rates between datasets that are non-spatial, non-time series, and have heterogeneous inputs. One dataset with these properties is credit card default data, which contains each client's age, education, etc., and whether they defaulted on their credit card. A dropout rate of 0.5 is widely used but does not always optimize performance. For each dataset, deep neural network models were trained over various dropout rates and training-set sizes. The experimental results presented here show that the optimum dropout rate falls anywhere within its possible range from 0 to 1, that even 10% dropout can significantly improve performance over no dropout, and that dropout can be effective even on small datasets.

Introduction

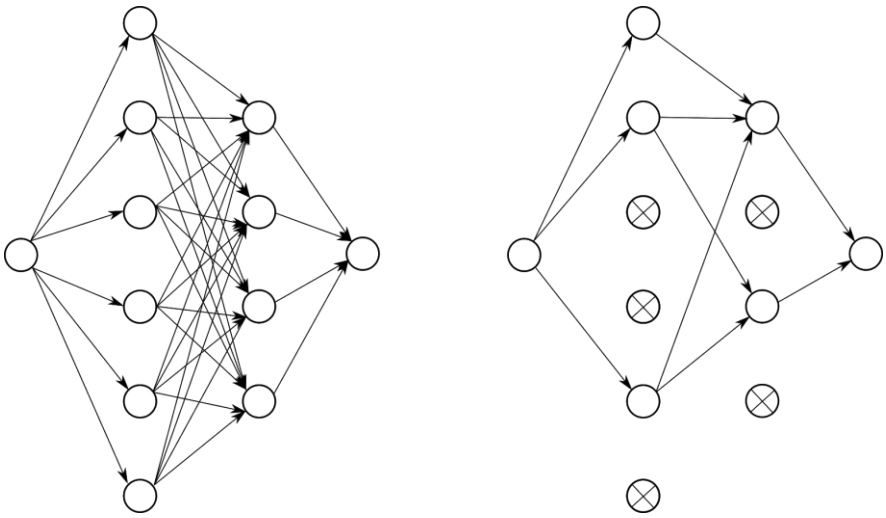
Data is more available than ever before. However, basic data analysis does not often result in accurate predictive models. To model the complex patterns within many of today's datasets, scientists often apply neural networks and other machine learning algorithms to detect complex nonlinear relationships in a dataset. One drawback to expressive networks is overfitting. Deep neural networks are very expressive, but that expressiveness allows them to overfit the training data, learning subtle distinctions that are merely artifacts of some particular training set. Figure 1a shows an example of overfitting. The samples deviate from the true function, or signal, because of noise, an effect that is common in real world applications. The overfitting model shown by the contour lines fits so well to the data that is also captures the noise. Such a model would perform extremely well during training but would perform poorly during testing or live implementation. A more versatile and useful model is shown in Figure 1b.



(a) (b)

Figure 1: An example of an overfitted model. (a) A neural network trained with 100 nodes on the PAY0 and PAY2 fields of the credit card default data set. (b) A neural network trained with 3 nodes. The contour lines show decision surfaces at different thresholds of the network.

One technique to reduce overfitting is dropout. The dropout technique avoids overfitting by dropping out different random nodes during training (Figure 2).



(a) (b)

Figure 2: The application of dropout to a neural network (a) A network without dropout (b) One iteration of training the network with a dropout of 50% on the hidden layers.

The contribution(s) of this work are:

- The effect of varying the dropout rates on datasets that are non-time series, non-spatial, and consist of heterogeneous inputs is studied for three datasets.
- There is not an optimum dropout rate shared by the datasets with similar properties. The experiments show that the optimal dropout rate can be anywhere within its range from 0 to 1, influenced by both the dataset selected and how many training samples are used from the dataset.
- While previous publications show that high dropout rates hurt performance of models trained on small datasets (Srivastava 2014), the experiments presented here show that dropout can improve performance even with a training set as small as 450 samples.

Background

Artificial neural networks (ANN) are one type of machine learning model. Similar to the human brain, ANNs contain a network of nodes, or neurons, that are interconnected. A deep neural network (DNN) is simply an ANN with multiple hidden “layers”.

A common obstacle with the application of neural networks is overfitting. Overfitting occurs when the network aligns too closely to the training data set. This leads to the network having a high predictive power with the training data set but a much lower success rate with the test data set or live data.

Dropout

Dropout is a technique for addressing this problem. The technique involves randomly dropping, or eliminating, neurons from the network during training (Srivastava 2014). This prevents units from co-adapting too much.

For each backpropagation, a new set of nodes is dropped out. At testing time, no dropout is applied. Because each backpropagation drops out so many nodes during training, the expressiveness of the model must be sustained by increasing nodes, layers, epochs, etc.

Following Baldi et. al (2013), this paper denotes the fraction of nodes dropped out during a backpropagation as q (this is the dropout rate), and the number of nodes remaining as p (this is the retention rate). This terminology helps to avoid confusion between the dropout rate and the parameter p, which both appear widely in the literature, but mean different things.

While dropout is most often applied to the hidden layers of a neural network, it can also be applied to the model's input nodes. This can reduce overfitting because the input layers can become redundant. With dropout, the model learns to consider redundant input nodes instead of relying on one. Dropout also increases the number of iterations required for the model to converge during training. For each epoch, a new random set of nodes is "dropped". Thus the model is consistently being trained with a lower number of nodes, requiring more iterations to converge. For example, using a dropout rate of q=0.5 roughly doubles the amount of iterations required to converge (Krizhevsky 2012). The model would train half the number of nodes for roughly twice the number of iterations. The additional computing time would come from the overhead computation time for each iteration. Thus, using a dropout rate of 0.5 increases computing time but by less than a factor of two. Along with adjusting the number of iterations during training, the weights must be adjusted during testing. For example, using a dropout rate of 0.5 during training requires the weights to be multiplied by 0.5 during testing.

The value of q=0.5 is often used for dropout. AlexNet, the network which achieved a step up in performance on the ImageNet classification challenge, used a dropout rate of 0.5 (Krizhevsky 2012). In a thorough review of dropout on very large problems, Srivastava commented that a dropout rate of 0.5 seemed to be close to optimal (Srivastava 2014). However, a similar image classification system using a deep neural network trained in MATLAB did not agree with the optimum dropout rate of 0.5 (Boddy 2017). For linear networks, a dropout rate of 0.5 provides the highest level of regularization (Baldi 2013). Most neural networks, however, are not applied to linear relationships. A contribution of this study is to demonstrate that the optimum dropout for a problem varies widely from one dataset to another and when a dataset's size is artificially reduced during training. Indeed, the optimum dropout could fall anywhere within the valid range for the parameter (0 to just short of 1).

ReLU: Rectified linear activation function

The rectified linear (ReLU) activation function is the most popular activation function for deep neural networks. It has an output range from 0 to infinity. It is 0 for $x < 0$ and is x for $x > 0$ (linear output). It reduces computation time over the softplus and sigmoid functions. (Krizhevsky 2012)

Producing a probability prediction with a Neural Network: The Softmax Function

The softmax function is used as the final layer of a neural network to produce a probability outcome instead of a classification. It essentially normalizes the sum of the values of the output layer. In the case of predicting the likelihood of credit card default, the softmax function would produce two values (default and non default probabilities) that add up to 1. The softmax function can be used in training through back propagation. (Krizhevsky 2012)

Experiments

Python was used to develop the deep neural network models. Specifically, the Python library TensorFlow was used to facilitate the model training and testing. Other libraries used include the Pandas and MatPlotLib.

The TensorFlow method DNNLinearCombinedClassifier and DNNLinearCombined-Regressor are used to train the models. Two hidden layers are used with the first consisting of 100 nodes and the second consisting of 50. Ten epochs are used for training the model. Furthermore, a variable representing the dropout rate is passed as an argument into the TensorFlow method used.

TensorFlow Accuracy Determination and Decision Threshold

The model accuracy is an output of the "evaluate" method within TensorFlow. It is calculated by first feeding the testing set through the trained model. The accuracy is then the number of correctly predicted classifications divided by the number of predictions made, or the size of the testing set.

The DNNLinearCombinedClassifier method within TensorFlow is rumored to use a default threshold of 0.5 when training the DNN. This can be changed by using the predict_proba method within DNNLinearCombinedClassifier to return a predicted probability for a given feature set, or sample. The return predicted probability can then be compared to a threshold and converted to the original binary classification.

Datasets used in our experiments: The Credit Card Default, Breast Cancer, and Bank Marketing Datasets

Due to their prominence, datasets that are non-time series, non-spatial, and have heterogeneous inputs are commonly used to create predictive models. The Taiwanese credit card default dataset (Yeh & Lien 2009) used in this study contains client attributes such as education level, marriage status, and past payments for approximately 30,000 clients. In addition, it includes whether or not each client defaulted on the credit card, which is necessary to train the model.

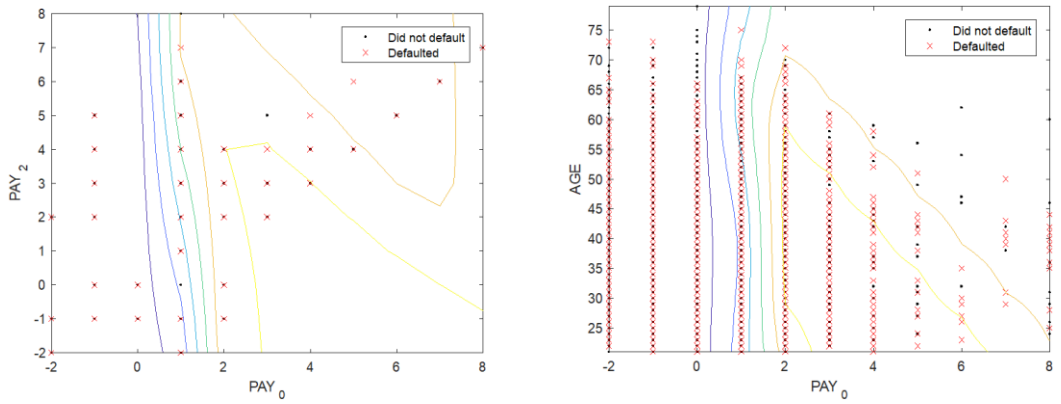
The breast cancer dataset (Wolberg & Mangasarian 1990) includes cancerous cell size uniformity, other diagnosis metrics, as well as the definitive diagnosis of whether a cancerous clump exists. This dataset consists of 699 samples.

The bank marketing dataset (Moro, Cortez, & Rita 2014) includes each clients financial background, marital status, amount of exposure of marketing campaigns, etc. and whether or not they subscribed to a term deposit. Models trained on this dataset contained a training size of 28,000 samples and a testing size of 15,000 samples.

Visually evaluating the separability of the credit card default dataset

A separable dataset is one which can be classified perfectly — a surface exists in the feature space which divides the categories perfectly.

To evaluate the separability of the credit card default dataset visually, a neural network was trained on just three of the input variables: age, pay0, and pay2 (The dataset does not include a pay1 field). The attributes pay0 and pay2 each represent the payment status of a month before the possible default. Negative values represent payment that was on time, and positive values represent the number of months a payment was late. Figure 3 shows the trained network's decision surfaces with various thresholds. The credit card data is far from separable when only these inputs are considered. There are many points with exactly the same inputs but both default and non-default outputs. Pay0 is a much stronger indicator of default than pay3 or age; the vertical decision lines show that changing the second variable does not change the prediction significantly.

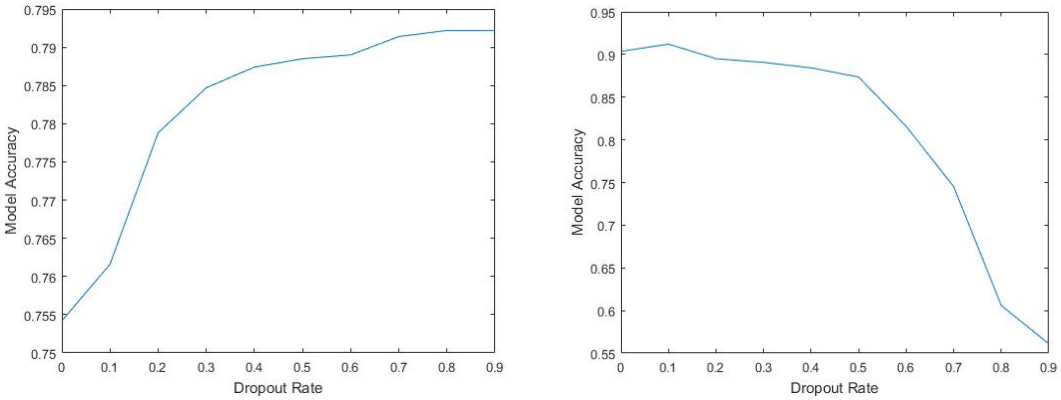


(a) (b)

Figure 3: The credit-card dataset is not separable when considering age, pay0, and pay1. A three-hidden node network trained on two inputs from the credit card dataset. Red X's indicate clients who defaulted and black dots those who did not. The red values are slightly offset to avoid overlap. The decision surface is shown in blue. The side to the right of decision surface represents clients predicted to default (the side with higher pay0 values). Two experiments are shown with different inputs: (a) pay0 vs pay2 and (b) age vs pay0

Evaluating the optimum dropout rate of different datasets

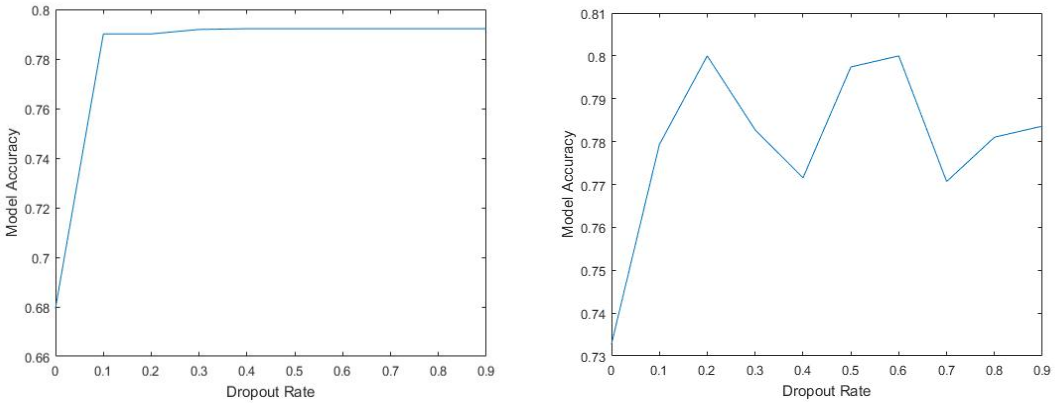
Figure 4(a) shows the accuracy of a model trained on the credit card default dataset that contains 18,500 samples. Figure 4(b) corresponds to a model trained on a breast cancer dataset that contains 450 samples. Both models were trained using two hidden layers consisting 100 nodes and 50 nodes. As can be seen, an increased dropout rate improved the model accuracy with a large dataset but decreased the accuracy with a considerably smaller dataset.



(a) (b)

Figure 4: Impact of dropout rate on a large and a small dataset (a) The credit card dataset containing 18,500 samples in the training set (b) The cancer dataset containing 450 samples in the training set.

To attempt to replicate the small training size effect seen in Figure 4(b) on the credit card default dataset, models were trained using 450 credit card clients across different dropout rates. Figure 5(a) shows the model accuracies that were determined with the previously used testing set of 1000 samples. As can be seen, the accuracy increases to an approximate maximum when any amount of dropout is present. In an attempt to keep parameters consistent between datasets, the 450 credit card training set was also paired with a testing set of 230 samples, which is the testing set size used in the breast cancer dataset. Five trials of this configuration were run across the dropout rates, and the averaged resulting model accuracies are shown in Figure 5 (b).



(a) (b)

Figure 5: Impact of dropout rate on a credit card training set of 450 samples while using large and small testing sets (a) The credit card dataset containing 1000 samples in the testing set (b) The credit card dataset containing 230 samples in the testing set

Much of the variation shown in Figure 5(b) is not repeatable from one set of trials to another. Figure 5(b) averages the results of five different trials. Figure 6 shows the first four trials that were used to construct the averaged curve in Figure 5(b). Figure 6 shows that the model accuracy is inconsistent between trials. This negates the confidence of possible conclusions drawn from Figure 5(b). The credit card default dataset likely has too

many variables and complexity for a 230 sample testing set to consistently gauge the model accuracies. Furthermore, the occurrence of default is relatively uncommon. Therefore, the randomly picked 230 sample testing set can range significantly in its proportion of defaulting clients.

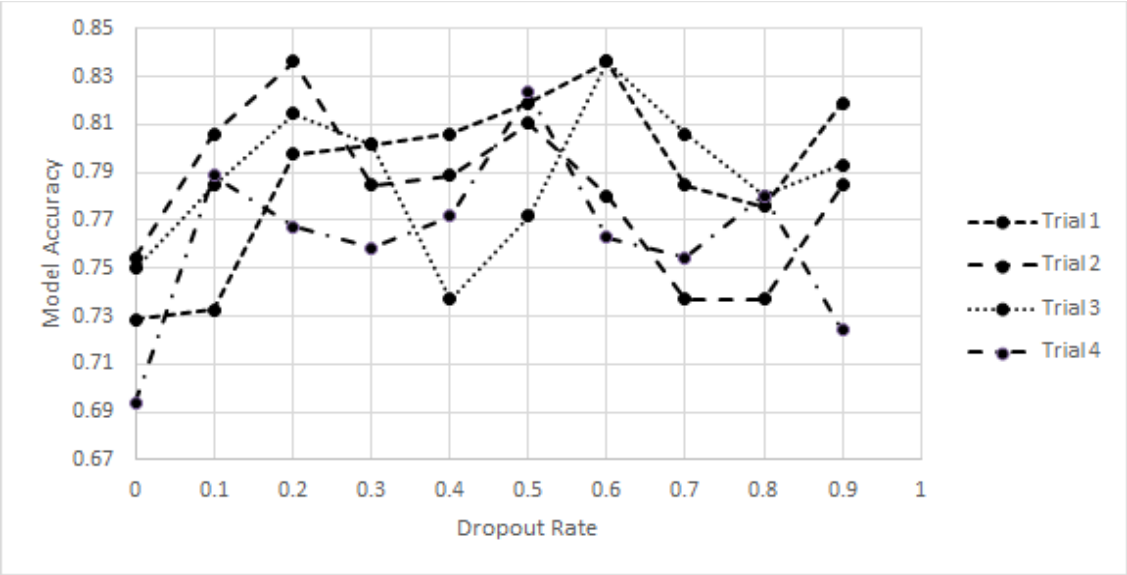


Figure 6: Different trials for the averaged data seen in Figure 5(b).

Figure 7 shows that the model performance for the bank dataset is not significantly affected by the dropout rate. It is difficult to determine an optimum dropout rate with a high confidence due to the small change in performance, which is less than 0.5%. If models trained on such datasets are not affected by the dropout rate, then no optimum dropout rate can be determined. Thus, it is not possible for such datasets to share an optimum dropout rate.

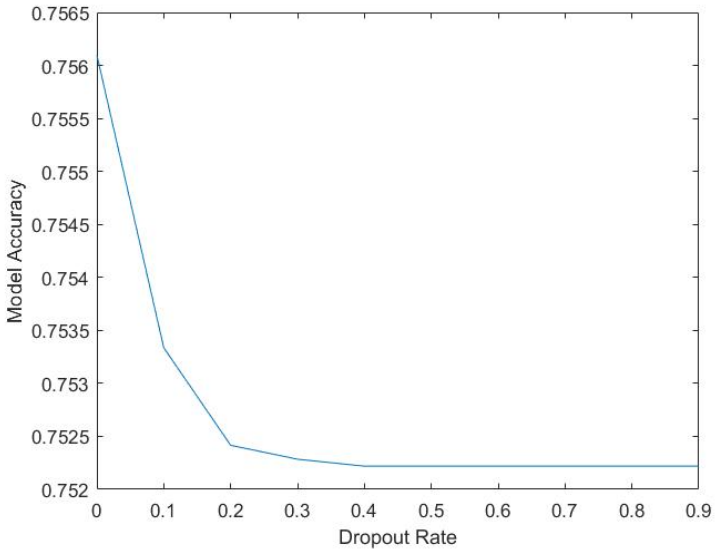


Figure 7: Performance of model trained on a 28,000 sample bank marketing dataset

Related Work

Although the specific form of dropout that we experimentally study here appeared in the literature only recently, the idea of dropping out nodes or edges has been in the literature for some time. For example, In the 1990's genetic algorithms were used to learn which nodes belonged in a neural network (Ronald and Schoenauer, 1994).

Dropout is considered as a sort of bootstrap aggregation or bagging technique (Breiman 1996) in which multiple models are trained on subsets of the data and then combined. Unlike the most straight-forward bagging implementation, all of the models share weights even though they have different structures at each step of training due to dropout (Krizhevsky 2012).

This work has emphasized the relationship between overfitting and neural network expressivity. Curiously, neural networks have a built-in ability to avoid overfitting even when they are capable of memorizing the input set (Zhang et al. 2016). It is not clear how this finding relates to the current work.

Conclusions

The optimum dropout rates for the credit card default, breast cancer, and bank marketing datasets were not consistently similar. Furthermore, the dataset size seemed to have a large effect on the optimum dropout rate, with smaller datasets performing better with low dropout rates. Due to the variance in optimum dropout rates for the studied models, the implementation of a universal dropout rate is not recommended. It is likely there are too many varying factors between different datasets that prohibit a common optimum dropout rate. Therefore, it is recommended that for each application of the deep neural network studied, the dropout rate be optimized before live implementation of the model.

Future Work

We plan to expand our experiments to include a larger variety of datasets. We plan to further explore the parameters that could influence the optimum dropout rate, including dataset size, number of features, number of hidden nodes, and separability.

We would like to consider the number of hidden nodes in addition to dropout rate and dataset size. This is important because the dropout rate affects the number of hidden nodes used during training. We would expect that increasing the number of hidden nodes would have a similar effect to reducing the dropout rate, except when the dropout rate gets close

to zero. Varying the number of hidden nodes also allows us to control the expressiveness of the neural network and provides an alternative way to avoid overfitting.

Another parameter we could explore is the number of hidden layers used during training. This parameter was not varied in this study because the network cannot be trained with more than approximately three hidden layers without becoming too expressive.

Bibliography

Baldi, Pierre, and Peter J. Sadowski. "Understanding dropout." *Advances in Neural Information Processing Systems*. 2013.

Bianchini, Monica, and Franco Scarselli. "On the complexity of neural network classifiers: A comparison between shallow and deep architectures." *IEEE transactions on neural networks and learning systems* 25.8 (2014): 1553-1565.

Breiman, Leo. "Bagging predictors." *Machine learning* 24.2 (1996): 123-140.

Boddy, Nicholas. *Object Classification using Deep Convolutional Neural Networks.* Midwest Instructional Computing Symposium (MICS) 2017

Breiman, Leo (1996). "Bagging predictors". *Machine Learning*. **24** (2): 123–140. doi:10.1007/BF00058655.

Hersbach, Hans. "Decomposition of the continuous ranked probability score for ensemble prediction systems." *Weather and Forecasting* 15.5 (2000): 559-570. http://journals.ametsoc.org. 2000.

Hinton, G.E., Krizhevsky, A., Srivastava, N., Sutskever, I., & Salakhutdinov, R. (2014). "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, 15, 1929-1958.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in Neural Information Processing Systems*. 2012.

Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Mangasarian, O. L. and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, 23.5, (1990): 1-18.

Ronald, Edmund, and Marc Schoenauer. "Genetic Lander: An experiment in accurate neuro-genetic control." *International Conference on Parallel Problem Solving from Nature*. Springer, Berlin, Heidelberg, (1994).

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research* 15.1 (2014): 1929-1958.

Yeh, I-Cheng, and Che-hui Lien. "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients." *Expert Systems with Applications* 36.2 (2009): 2473-2480.

Zhang, Chiyuan, et al. "Understanding deep learning requires rethinking generalization." *arXiv preprint arXiv:1611.03530*(2016).