5-2017

Investigation of How Neural Networks Learn From the Experiences of Peers Through Periodic Weight Averaging

Josh Reeves Smith
University of Arkansas, Fayetteville

Investigation of How Neural Networks Learn From the Experiences of Peers Through Periodic Weight Averaging

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

by

Joshua Smith
University of Arkansas
Bachelor of Science in Computer Engineering, 2015

May 2017
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

_____

Dr. Michael S. Gashler
Thesis Director

_____

Dr. Susan Gauch
Committee Member

_____

Dr. Wing Ning Li
Committee Member

Abstract

We investigate a method, weighted average model fusion, that enables neural networks to learn from the experiences of other networks, as well as from their own experiences. This method is inspired by the the social natural of humans, which has been shown to be one of the biggest factors in the development of our cognitive abilities. Modern machine learning has focuses predominantly on learning from direct training, and has largely ignored learning through social engagement with peers, neural networks will the same topology. In order to explore learning through engagement with peers, we have created a way for neural networks to teach each other. Our method allows neural networks to exchange knowledge by combining their weights. It calculates a pairwise weighted average of the weights of two neural networks, and then replaces the existing weight with the new value. We find that weighted average model fusion successfully enables neural networks to learn from the experiences of their peers and combine it with the knowledge that is gained from its own individual experiences. Additionally, we explore the effects that several meta-parameters have on model fusion to provide deeper insights into how the behaves in a variety of scenarios.

Acknowledgments

I would like to thank my mother, Karen, and my father, Larry, for their continued support throughout my educational career, as well as all of my friends, who have made the experience that much more enjoyable. I would also like to thank Dr. Gashler, who has continually pushed me to exceed my own expectations, and my committee advisors, whose advice and experience has been vital in the success of this project. Finally, I would like to thank the University of Arkansas Computer Science and Computer Engineering department and the Graduate School for supporting my education and research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One thing that makes humans unique is our exceptional cognitive abilities. Studies have shown that the "ultra-social" nature of humans has been a major influence in the development of human intelligence [12]. However, in machine intelligence methods for exchanging knowledge between peer machine learning models have not been widely investigated. Most machine learning techniques learn by first-hand pattern presentation; making observations about the data and refining their internal representations over time. In this paper we introduce a method for fusing neural network models together, and we validate its effectiveness for learning from second-hand experiences. This method can enable neural networks to learn indirectly from each others' experiences, while still achieving comparable levels of accuracy to that of a single model trained by all of the data.

Neural networks are powerful machine learning models that can theoretically approximate any arbitrary function [14]. Iterative weight training methods, such as stochastic gradient descent [2], AdaGrad [8], RMSprop [21], Adam [15], have been generally effective at refining models to fit training data. By altering the weights, these methods slowly refine the function that can accurately describe the training data and make more accurate predictions. So the weights store the knowledge of the neural network. Our approach is based upon this idea that by iteratively combining the weights of two networks, at certain intervals during the training process, we can effectively pass knowledge between two neural networks. This is most applicable in scenarios where data is distributed.

For example there are over five thousand hospitals in America and even more throughout the world. Each one has a large storage of patient information, which is often of a personal and sensitive nature. Mining that data is not currently feasible because existing models require the data be aggregated at a central location, and hospitals are not allowed to share their data due to restrictive privacy laws. Weighted average model fusion provides a way to learn without aggregating data.

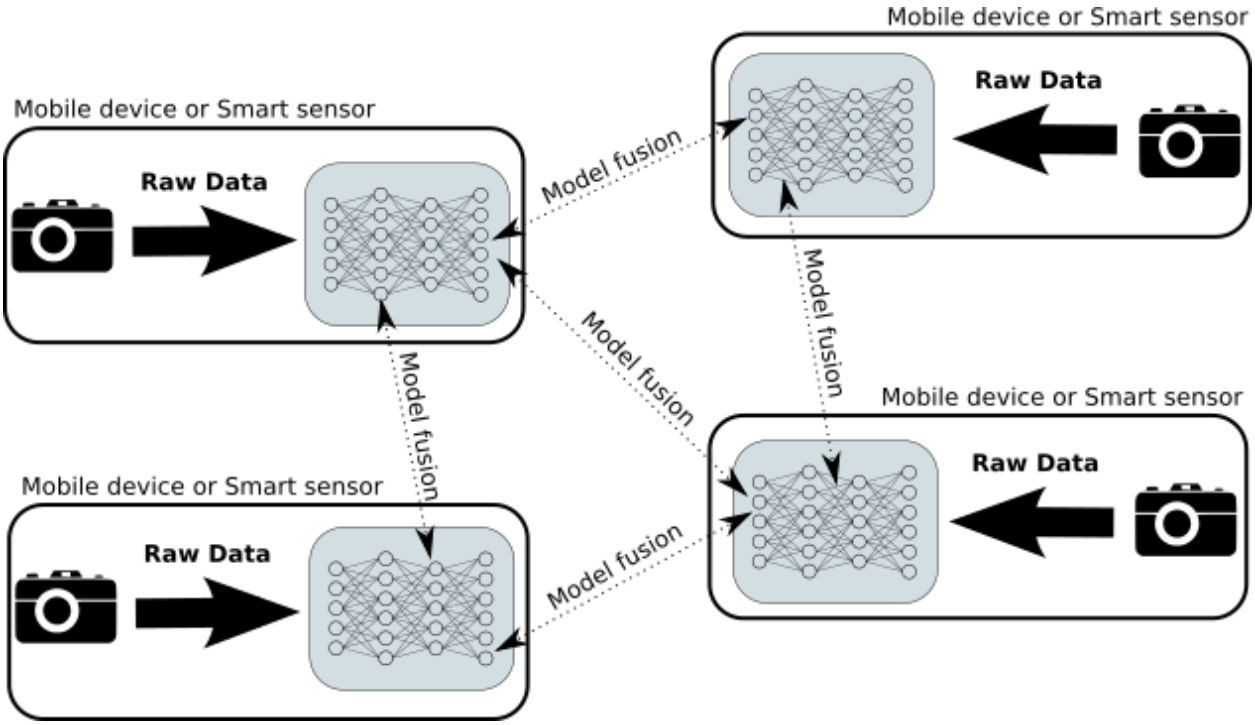

Figure 1.1: Smart Grid Model Fusion Example
Model fusion enables deep neural networks to benefit from training that has been applied to peer neural networks. The big raw data stays on the local device, while models (which are typically orders of magnitudes smaller) are exchanged between peers at infrequent intervals. The resulting neural networks have only seen a local portion of the raw data, yet perform as if they had been trained on all of it.

Another example of learning that requires the aggregation of data is smart grid technology, where devices are placed throughout a city. Using weighted average model fusion these devices can learn based upon the collective experience of all devices to more accurately perform their tasks. Additionally, the image processing models or navigation models of a self driving car can learn not just from its own experiences but from the experiences of a multitude of cars, without having to collect data from other cars. Our method can circumvent the data collection process and attack the problem directly. By combining the weights of neural networks these models can gain insights that, without data aggregation, they would not have. Bypassing data aggregation is a benefit of machine learning models have the capability of exchanging insights gained from their individual experiences.

Allowing neural networks to exchange knowledge with one another draws inspiration the social

nature of humans, which has been shown to be a significant aspect of how humans learn. A study done in 2007 determined that while some species of apes behave socially or culturally, these interactions are not their primary way of learning [12]. That same study performed tests on a 2 year old human child, before it has had any form of formal education, a chimpanzee, and an orangutan. On physical cognition tests, all three participants performed similarly, but on social cognition test the human child performed considerably better. These results demonstrate the innate ability that humans have to learn through social interactions. There is evidence to suggest that these social interactions were the driving force behind the evolution of human intelligence [4].

This type of learning is very powerful. Due to the significant impact it has on human intelligence, we believe that learning through social interactions merits investigation as an approach to machine learning. In this paper we introduce a new technique, that enables neural networks to learn from one another, and explore the effects of its various parameters.

Thesis Contributions

To support these claims, this this provides the following contributions:

1. We present a method of combining neural networks using a pairwise weighted averaged of the weights of the neural network.

2. We present three different fusion topologies that describe the manner in which pairs of neural networks are fused together.

3. We demonstrate that our method for allowing neural networks to learn from each other can predict with similar accuracy to that of a single model trained by all of the data.

4. We define the meta-parameter the fuse rate and explore the effects it has on weighted average model fusion.

5. We define the meta-parameter the frequency and explore the effects it has on weighted average model fusion.

Organization

The remainder of this thesis is organized as follows:

Chapter 2 provides an overview of the related work. In Chapter 3, we describe the method, weighted average model fusion, that allows neural networks to exchange knowledge. We also detail three separate fusion topologies: Random, Ring and Hypercube. Next in Chapter **??**, the methods for model fusion and fusion topologies are thoroughly tested and evaluated. We also investigate how the meta-parameters of model fusion affects the prediction accuracy of the resulting neural networks. Finally, we conclude in Chapter 5.

Chapter 2

Background

There have been many algorithms developed with the goal of combining neural networks together. The most well established approach to this problem is ensembles. There have been many ensemble techniques that have been developed over the years that combine the results of neural networks. Ensembles of neural networks originated from Hansen and Salamon's work [11]. Since then there have been numerous studies done that build on their work, Multiple Networks Fusion using Fuzzy Logic [6], Ensembling neural networks: Many could be better than all [23], Neural Network Ensembles, Cross Validation, and Active Learning [16], Design of effective neural network ensembles for image classification purposes [10], Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction [20] and Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles [17].

Ensemble techniques combine the output of each neural network in a variety of different ways. There has been a fair amount of research that has gone into developing the best approach of combining the outputs of multiple machine learning models together. However, this is a direct contrast to our approach. Our technique takes the weights of neural networks and intelligently combines them together to improve the internal representation of both networks. Allowing both networks to make more accurate predictions as opposed to having multiple network make prediction, and leveraging multiple prediction to enhance accuracy. Ensemble techniques are also not sufficient to address the hospital problem described in the previous section because the insights into how much each input features determines a specific output can be leverage by combining the outputs themselves. The impacts these features have are saved in the weights of the network. So by combining the weights the neural network will have a more robust internal representation of the problem.

There have been several other approaches that allow neural networks to train other networks. Caruana and his collaborators first showed that it was possible to compress the knowledge of an

ensemble into a single mode[3]. In addition to their work researchers at Google have developed similar algorithms [13]. Deep networks have been shown to be able to handle much larger and more complex tasks than shallow networks. The algorithms presented in these papers both use a deep network to train a shallow network. Showing that shallow networks have the potential to perform just as well on complex tasks.

There is another algorithm for transferring knowledge between neural networks called Net2Net [5]. Net2Net uses an existing teacher network to train a student network. Unlike other algorithms that compress the model [13] [3] [5], this approach expands the model. They have two separate algorithms: one for expanding the number of weight within a layer and one for expanding the number of layer in a network. Both algorithms take nodes or layers from a teacher network and incorporate them into a student network. This change in the topology makes the training process faster because some of the weights, in the layer, already have values. Unlike our technique, these methods involve a retraining process. Our technique fuses networks together as they are trained, so the learning enhancements occur at training time.

There has also been work on parallelizing stochastic gradient descent for training deep networks. An algorithm called HogWild [18] calculates the gradients of parallel networks and combines the gradients of each network to update the centralized neural network. Another algorithm called Elastic Averaging SGD [22] finds a running average of parallel networks as an update rule. These methods are similar to our approach, but they are an attempt to parallelize training of a centralized model to decrease training time. Our approach seeks to train distributed models to demonstrate the effectiveness of neural networks learning from each other.

There has also been work done with averaging the weights of single-layer perceptrons [1], referred to as "Wagging". "Wagging" seeks to improve predictive accuracy for a single layered perceptron. We seek to enable applications that require training of multilayer perceptrons, without aggregating the data.

Chapter 3

Methods

When training a neural network an optimization method is used to refine the model by making changes to the weights.

Model Fusion

Algorithm 1 Weight Averaging Model Fusion

function Model Fusion(N_1, N_2, F)
  A = weights of N_1
  B = weights of N_2
  let W be the size of A (which also the size of B).
  for $w \in W$ **do**
    if A_w == 0 **then**
      A_w = B_w
    else if B_w == 0 **then**
      B_w = A_w
    else
      A_w = (1.0 - F)*A_w + F*B_w
      B_w = (1.0 - F)*B_w + F*A_w
    end if
  end for
  weights of N_1 = A
  weights of N_2 = B
end function

In order two fuse two networks together we combine the weights by calculating a pairwise weighted average of those weights, as described in Algorithm 1. So for two consecutive networks we extract the weights, iterate through them, comparing them to their counterparts in the other network. For each weight in the network we calculate a weighted average using a meta-parameter called the fusion rate or fuse rate. During the fusion process, the fusion rate determines which network will be more heavily favored in the fusion.

The new weight is the sum of two products: one minus the fuse rate multiplied by the current

weight and the fuse rate multiplied by the respective weight in the other network. This is repeated for both networks. The fusion rate is the weight we assign to a network when we fuse. Given two networks A and B, where A is the resulting network of fusion thus far and B is the network we need to incorporate into the fusion, a low fusion rate favors the knowledge from the new network and a high fusion rate favors knowledge from the fusion network. Apply a fusion rate to zero weights would skew the internal representation of the inputs. Some forms of regularization would cause the weights to be exactly zero and in these scenarios we want the non-zero weight to prevail. This algorithm is repeated at set intervals during the training process.

Fusion Topology

The Fusion topology is the structure of the network of machine learning models, specifically neural networks (as opposed to the topology of a neural network itself). These algorithms iterate through a set of neural networks and perform weighted average model fusion of pairs of neural networks. We have evaluated three different fusion topologies: ring fusion topology, random fusion topology, and hypercube fusion topology.

Ring Fusion Topology

Algorithm 2 Ring Fusion Topology

function Ring Fusion
  let N be a list of neural networks.
  let S be size of N.
  let F be the fusion rate between 0 and 1.
  for $s \in \{0, \ldots, S\}$ **do**
    if s+1 <S **then**
      ModelFusion(N_s, N_{s+1}, F)
    else
      ModelFusion(N_S, N_0, F)
    end if
  end for
end function

The ring fusion topology, as described in Algorithm 2, is similar to a ring network. Each neural

network is connected to exactly two other networks, which forms a continuous path, a ring. Given a set of neural networks, N, we iterate through the set combining each network to it's neighbor. So N_0 and N_1 fuses, then N_1 and N_2 fuse, until it comes to the end of the list. Once we reach the end of list we fuse the last network with the first completing the circle. This accumulates the knowledge of the each network a the fusion moves around the ring, eventually end back at the starting network. The starting network will then have the cumulative knowledge of all neural network in the ring.

Random Fusion Topology

Algorithm 3 Random Fusion Topology

function Random Fusion
  let N be a list of neural networks.
  let S be size of N.
  let F be the fusion rate between 0 and 1.
  let R_1 and R_2 be uniformly distributed between 0 and S.
  while $S \geq 2$ **do**
    $r1$ is a random value between 0 and S.
    $r2$ is a random value between 0 and S.
    ModelFusion(N_{r1}, N_{r2}, F)
    $S = S - 2$
  end while
end function

The random fusion topology, as described in Algorithm 3, consists of a set of neural networks, where random connections are made between networks in every time model fusion is performed. Given a set of neural networks, N, we make random connections between neural networks where every network is connected to another random network. These connections are reset every time fusion is performed. So as fusion is performed different pairs of networks are learning from each other and over time every network will communicate with every other network. This will allow the cumulative knowledge to stochastically be passed around over time.

Hypercube Fusion Topology

Algorithm 4 Hypercube Fusion Topology

```
function Hypercube Fusion
    let N be a list of neural networks, that is represented a n bit binary number.
    let S be size of N.
    let F be the fusion rate between 0 and 1.
    for x ∈ {0, ..., n} do
        for s ∈ {0, ..., S} do
            b = binary number of N_s, with the x bit inverted.
            ModelFusion(N_s, N_b, F)
        end for
    end for
end function
```

The hypercube topology is based upon the hypercube graph. A hypercube graph, also called a n-cube graph, that consists of 2^n vertices and $2^{n-1}n$ edges. A 2-cube graph would be a square and a 3-cube graph would be a cube. Hypercube graphs have been proven to be very power topologies [19] [7]. One of the more attractive properties of a hypercube is its small diameter. Diameter is the maximum number of links between any vertices of a graph, for hypercubes the diameter is n. For model fusion we pass defines each vertex of this graph is a neural network.

Each neural network is combined with each of its neighboring neural networks. So for a n-cube graph, each network is fused with n neural networks, as described in Algorithm 4. This is seeks to take advantage of the small diameter that exists within hypercube graphs. The small diameter facilitates the quick transmission of knowledge between any two networks in the graph, which should allow the individual knowledge of each neural network to be distributed among it's peers evenly.

Chapter 4

Evaluation

In this section, we present empirical performance of our algorithm. We ran all tests on three different datasets: MNIST, Vowel, and Wisconsin breast cancer. For all of our experiments every network we used was a feed forward neural networks and the tanh activation function. These experiments were coded using C++ and Waffles machine learning library. [9]. We ran these experiments on 4-core Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz machine with 16 GB of RAM.

Model Fusion

We evaluated our new technique by running two experiments. For the first test we created a set of 10 neural networks and trained each of them all an individual digit from the MNIST dataset. During the training process we periodically combine the neural networks together so that all 10 networks can learn to recognize all ten digits. We compare the neural network that resulted from model fusion to a single neural network that had been trained on all ten digits and another single neural network that had only on images of a single digit.

We use these two neural networks to create ideal performance bounds for out algorithm. If the algorithm does not introduce any new knowledge to the networks being fused, then they will perform as if they had only been trained on images of a single digit. If the algorithm each network teaches all of the other networks to recognize it's assigned digit, then the any one of those networks will perform as well as a single neural network that has been trained on all of the digits. As shown in Figure 2, the network trained using images of just one digit achieved around 88 percent error and the network trained on images of all ten digits achieved around 3 percent error. The middle is the result of a collection of networks using the random fusion topology. It achieved comparable levels of accuracy to that of the network that has been trained on all of the data. This empirical demonstrates that model fusion effectively allows neural networks to learn from one another.

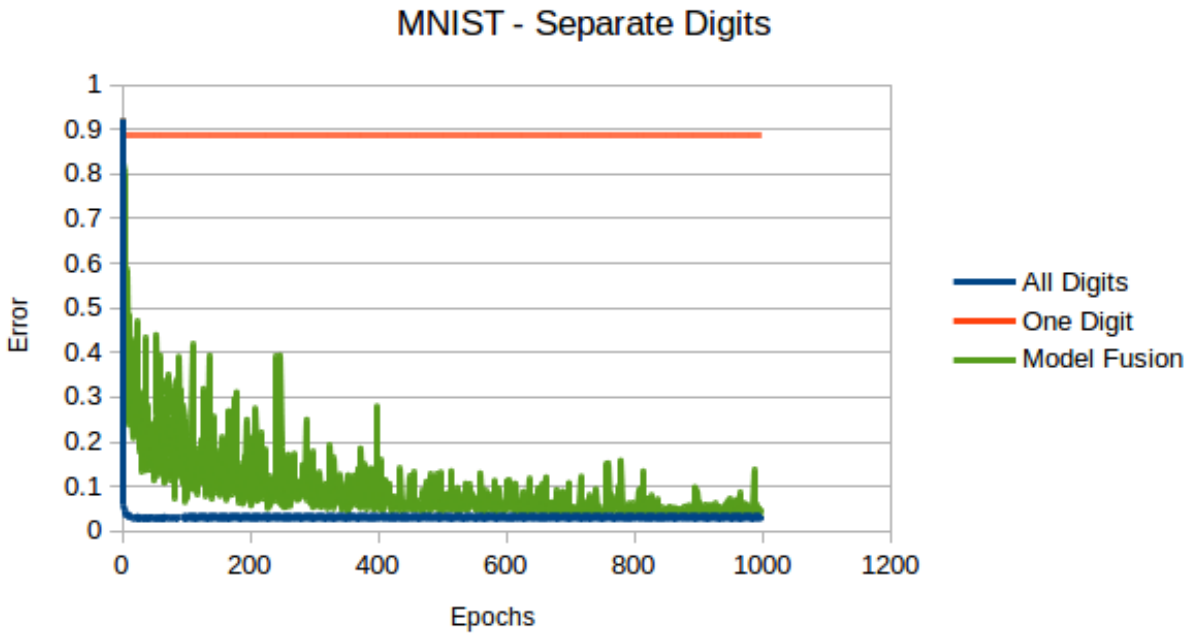

Figure 4.1: MNIST Separate Digit Results
Compares the percent error over time for a network that has trained on all of the digits, one that has trained on just on one of the digits, and the result of using the random fusion topology. All of the networks in this problem had a learning rate of 0.01 , two hidden layers of 80 and 30 nodes, fusion rate of .55, and a fusion frequency of 200 tuples.

Table 4.1: Table of predictions

Percent Error					
Dataset	Min	Max	Ring	Random	Hyper-cube
MNIST	0.0294	0.0666	0.0292	0.0334	0.0299
Vowel	0.3752	0.5011	0.3557	0.3687	0.3752
Segment	0.0317	0.0981	0.0418	0.0505	0.0389
Breast-W	0.00971	0.0291	0.00971	0.00971	0.0145

The second test we created a collection of 8 neural networks, where each network trained on a random subsample of the data. We compared the final results of Model fusion to a single neural network that has trained using all of the data and a single neural network that trained using an eighth of the data. As before, these two single neural networks serve as performance bounds that provide perspective on the performance of weighted average model fusion. Unlike the first test, these neural networks are only restricted by how much data they are given, not by what data they are given.

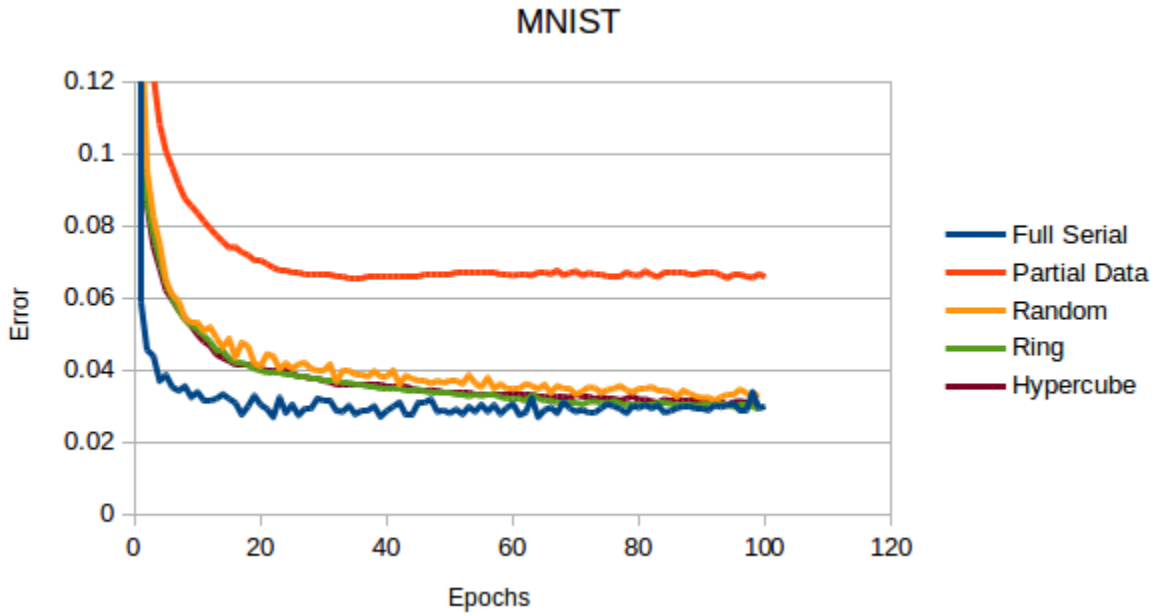

Figure 4.2: MNIST Model Fusion Results
Compares the percent error over time for a network that has trained on all of the data (All Data), one that has trained on part of the data (Partial Data), , and the result of different fusion topology techniques Random, Ring an Hypercube. All of the networks in this problem had a learning rate of 0.01 , two hidden layers of 80 and 30 nodes, fusion rate of .55, and a fusion frequency of one epoch.

For each of the problems we optimized the individual networks that we are using for comparison. So for each problem we performed grid search to find the optimal learning rate and chose an architecture that performed reasonably well using non-model fusion networks. Each network was trained using stochastic gradient descent. Varying topologies were used amongst the various problems. For mnist we used two hidden layers of 80 and 30 nodes, for the vowel and segment datasets we used two hidden layers of 50 and 20 nodes, and for breast-w dataset we used one hidden layer of 8 nodes.

We report the percent error and show how it decreases over time. This is the most direct approach in demonstrating how well each approach performs. In all 3 examples, model fusion achieves comparable accuracy as the individual network that was trained on all of the data. This is significant because each model in the model fusion case was limited to training with a small

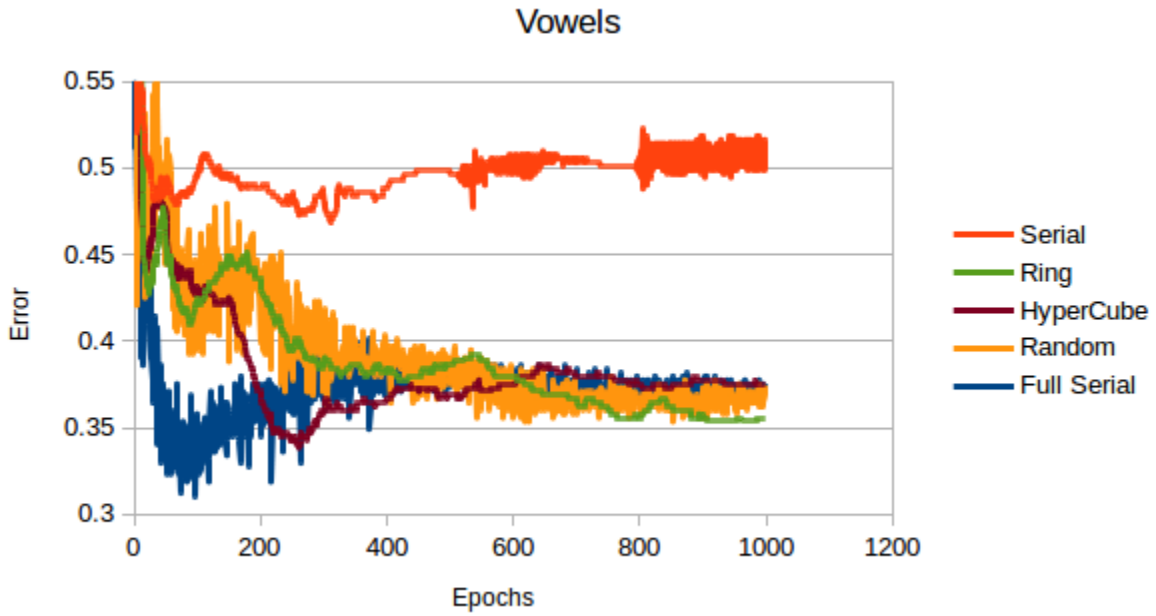

Figure 4.3: Vowel Model Fusion Results
Compares the percent error over time for a network that has trained on all of the data (All Data), one that has trained on part of the data (Partial Data), and the result of different fusion topology techniques Random, Ring an Hypercube. All of the networks in this problem had a learning rate of 0.04 , two hidden layers of 50 and 20 nodes, fusion rate of .55, and a fusion frequency a fusion frequency of one epoch.

portion of the data.

Figure 3 and Figure 4 report the results from the vowel and MNIST tests, respectively. In both cases Model Fusion takes more time to converge than the network that has trained on all of the data, but it does make predictions with comparable accuracy. When we tested Model Fusion on the vowel dataset the single model, that trained on all of the data, overfit to the data, decreasing in accuracy, but the model fusion did not. Table 1 reports the percent error for all four experiments. These results demonstrate how Model Fusion enables neural networks to obtain the same level of accuracy as a neural network trained on all of the data. By obtaining this level of accuracy it proves the weighted average model fusion will allow neural networks to teach each other. If model fusion had not worked the prediction accuracy would be closer to that of the network trained on a subset of the data. However, in every experiment weighted average model fusion achieved an accuracy

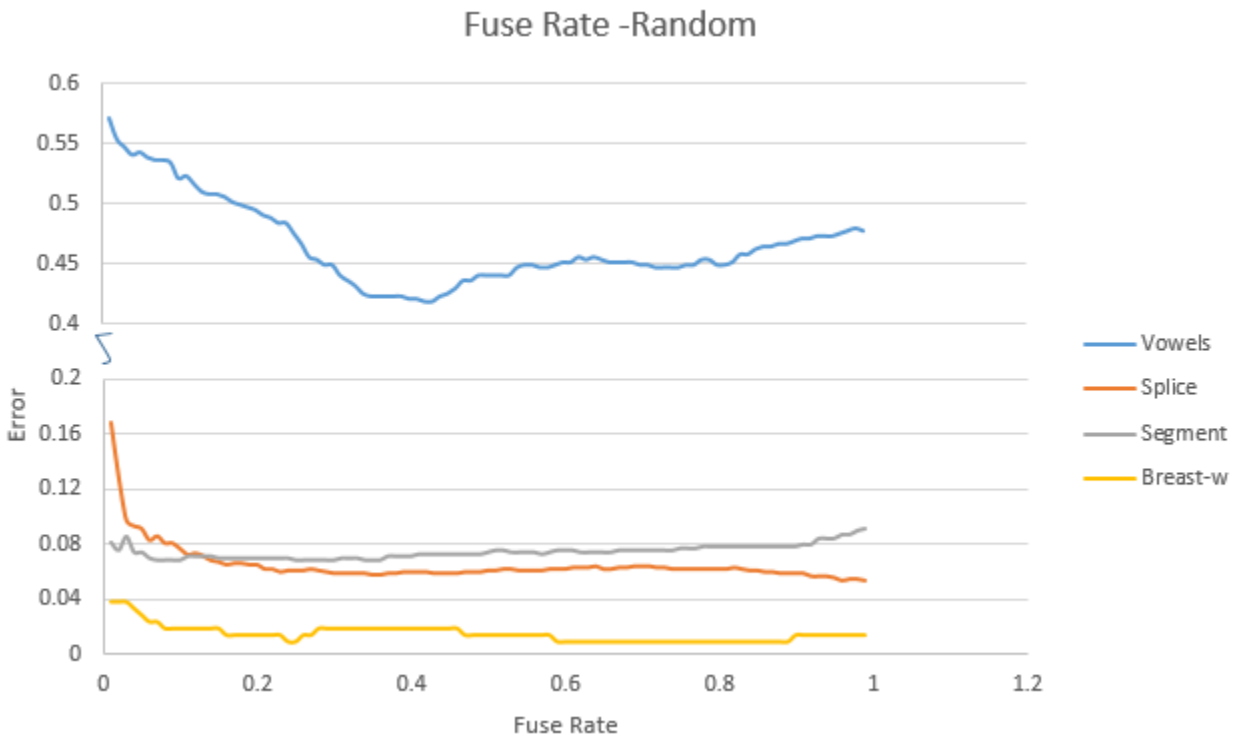

Figure 4.4: Exploring fuse rate using a Random fusion topology
Using the random fusion topology compares percent error as the fuse rate increases for the four dataset: vowel, segment, breast-w, splice.

within one percent of the network trained on all of the data.

Meta Parameters

Model Fusion has two meta-parameters: fusion rate and fusion frequency. In these experiments we demonstrate the effects each meta-parameter has on the accuracy of model fusion and therefore draw conclusions on the effects these parameters plan in the learning process.

Fusion Rate

In order to understand the effects of the fusion rate, we ran 4 tests on the datasets from the UCI classification dataset collection: vowel, breast-w, segment, and splice using all three fusion topologies. For each dataset we ran the fusion algorithms on 8 different networks. We compared the resulting accuracy of all fusion rates from 0.01 to 0.99, with a step size of 0.01.

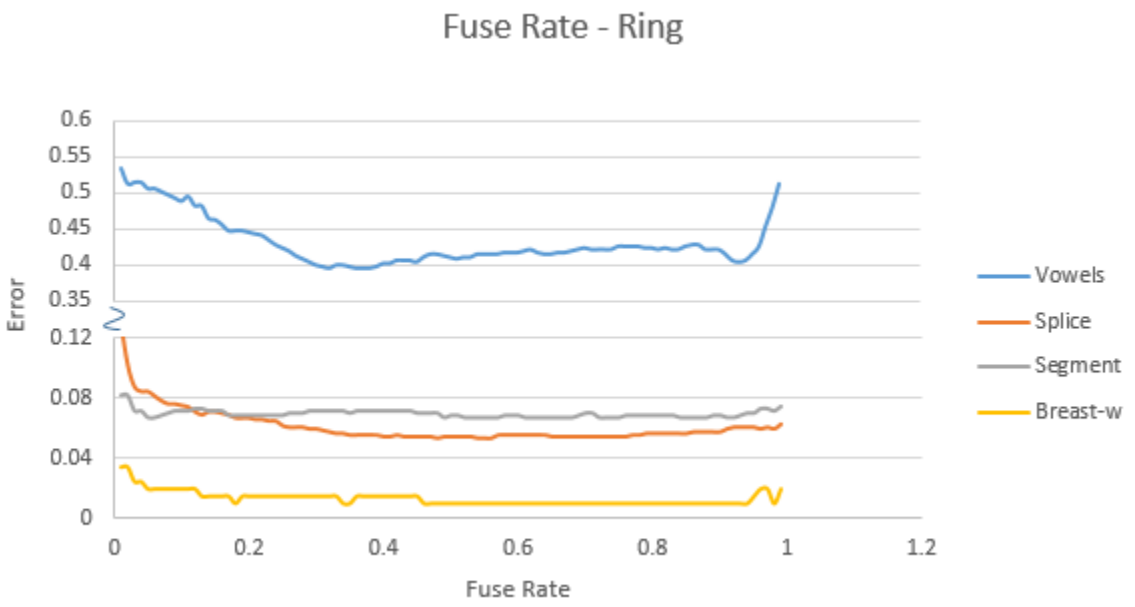

Figure 4.5: Exploring fuse rate using a Ring fusion topology
Using the ring fusion topology compares percent error as the fuse rate increases for the four dataset: vowel, segment, breast-w, splice.

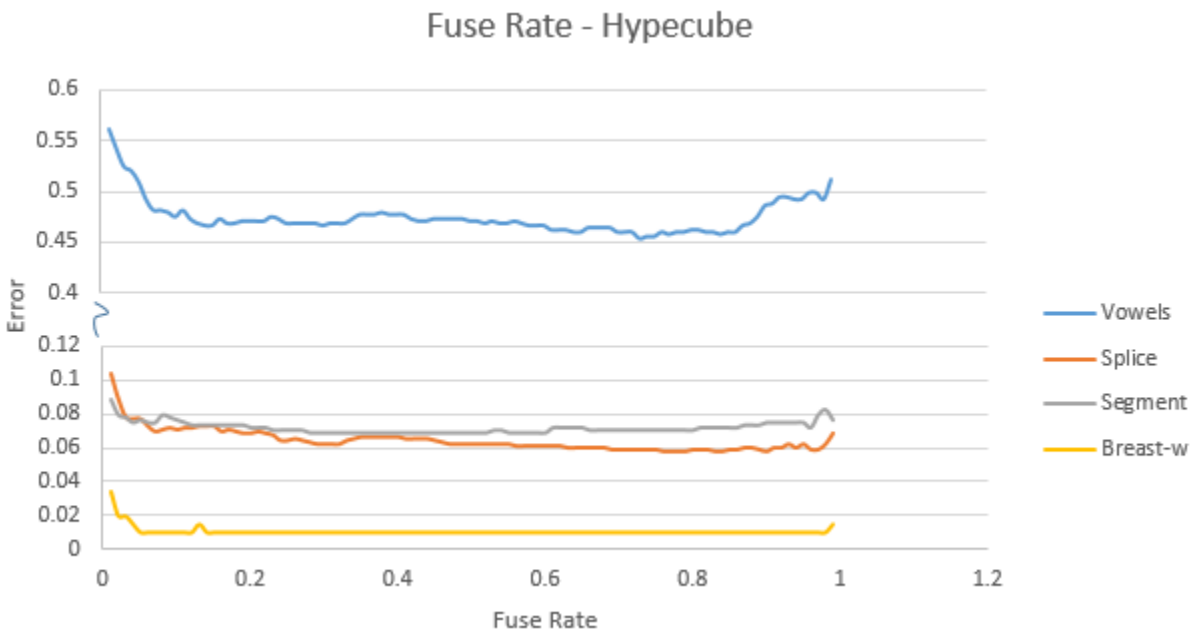

Figure 4.6: Exploring fuse rate using a Hypercube fusion topology
Using the hypercube fusion topology compares percent error as the fuse rate increases for the four dataset: vowel, segment, breast-w, splice.

For the experiments using the random fusion topology, the fuse rate seems to resemble that of the accuracy as a neural network trains. The higher the fusion the rate, the error decreases until a certain point where it levels off. For splice dataset the global minimum occurred when the fuse rate was 0.98. However, the next closest value were that between 0.30 and 0.38. The vowel tests reached a clear global minimum when the fuse rate equaled 0.42.

In the experiments using ring fusion topology and hypercube fusion topology. a similar trend held true. The fuse rates produces similar results after a certain value, reaching the best results for different fuse rates in all four tests. The fact that values under 0.2 never perform as well as values above 0.2 indicates that favoring one network too heavily works well. It biases the knowledge that is spread throughout the collection of neural networks so much that the accuracy is never able to converge. The best values for fusion rates consistently seem to be the rates just below 0.5. This means that fusion rates that favor the knowledge that individual network has learned over that of its peer produces more accurate results. This allows the network to gain the knowledge from the other networks while still maintain the bulk of what it has personally learned.

Fusion Frequency

The fusion frequency defines the interval at which the fusion algorithm occurs during the training process. This determines how much change occurs in the individual neural networks during the training process. In order to discover the affects this meta-parameter has on the accuracy of the fusion we conducted tests on 4 datasets from the UCI classification dataset collection vowel, breast-w, segment, and splice. We ran this test for all three fusion topologies. For each dataset we tested the values from 1 epoch to 100 epochs, with a step size of 1.

For all three types a fusion, the error increased the less frequently the fusion occurred. The experiments on the vowel dataset show a consistent trend of the larger the fusion frequency the less accurate the models become. However, this is not always true when viewing the trend at smaller intervals. There several small intervals in all three tests where increasing the fusion frequency decreased the error in the predictions.

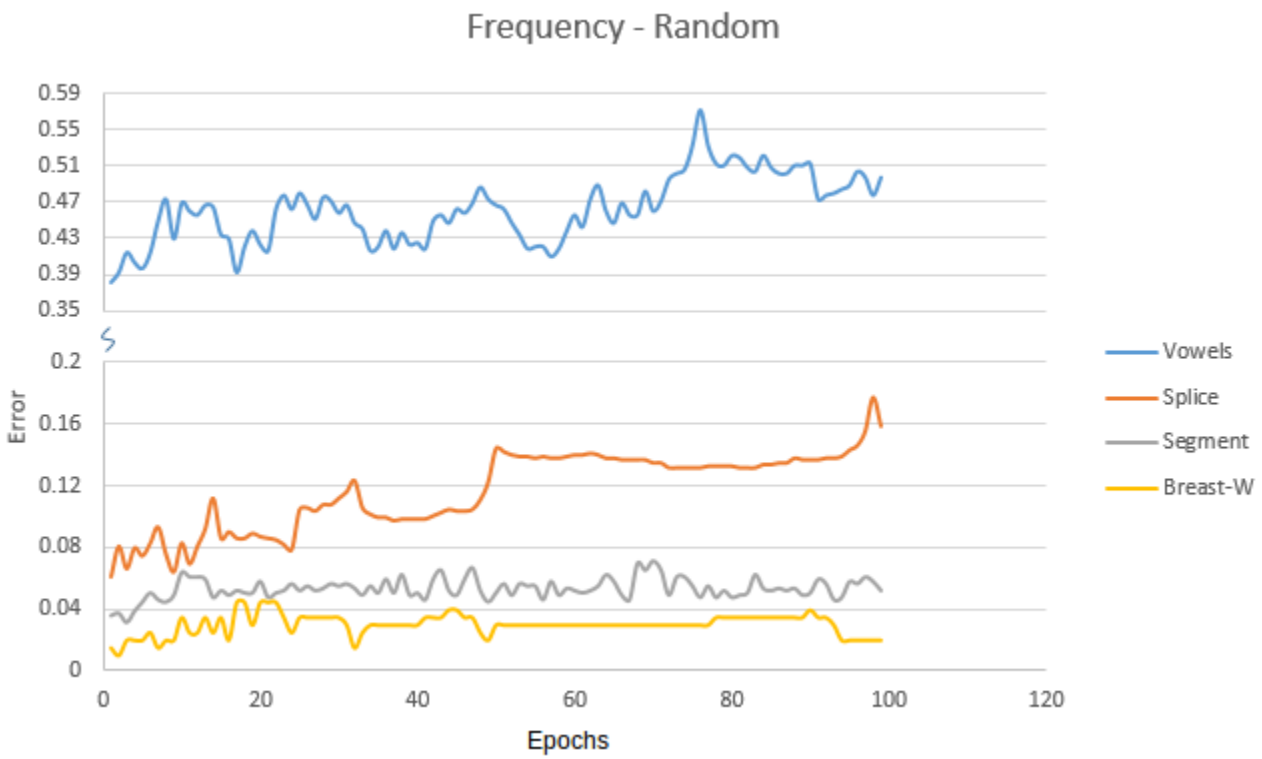

Figure 4.7: Exploring frequency using a Random fusion topology
Using the random fusion topology compares percent error as the frequency increases for the four dataset: vowel, segment, breast-w, splice.

For the other three datasets: splice, breast-w, and segment there sporadic changes happen as frequently as they do in the vowel test, but there is a less significant change in the overall accuracy as the fusion frequency increases. The splice dataset increased overall in all three experiments, but when using the hypercube topology it's increase was not very large. The breast-w dataset and segment dataset had sporadic changes, but overall were fairly consistent.

The sporadic changes that exist within all of our experiments suggest that the fusion frequency is problem dependent and is something that needs to be tuned much like a learning rate. Nevertheless the overall trend for more frequent fusions leading to more accurate results seems to hold true in all cases. So in choosing a fusion frequency, these results suggest that, lower values tend to produce more accurate results.

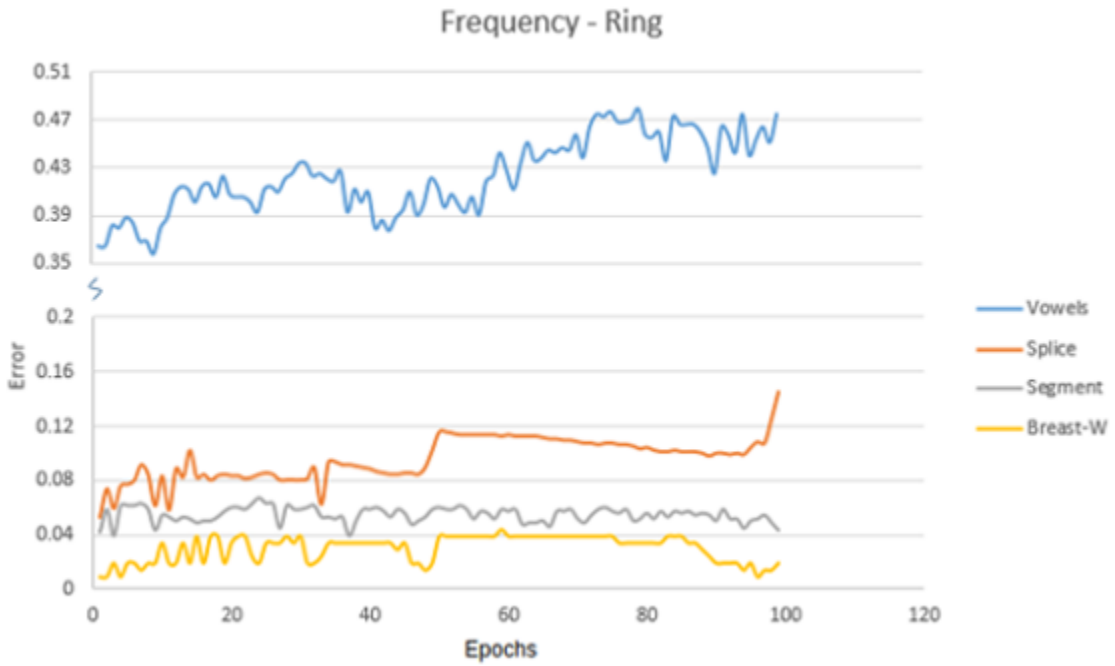

Figure 4.8: Exploring frequency using a Ring fusion topology
Using the ring fusion topology compares percent error as the frequency increases for the four dataset: vowel, segment, breast-w, splice.

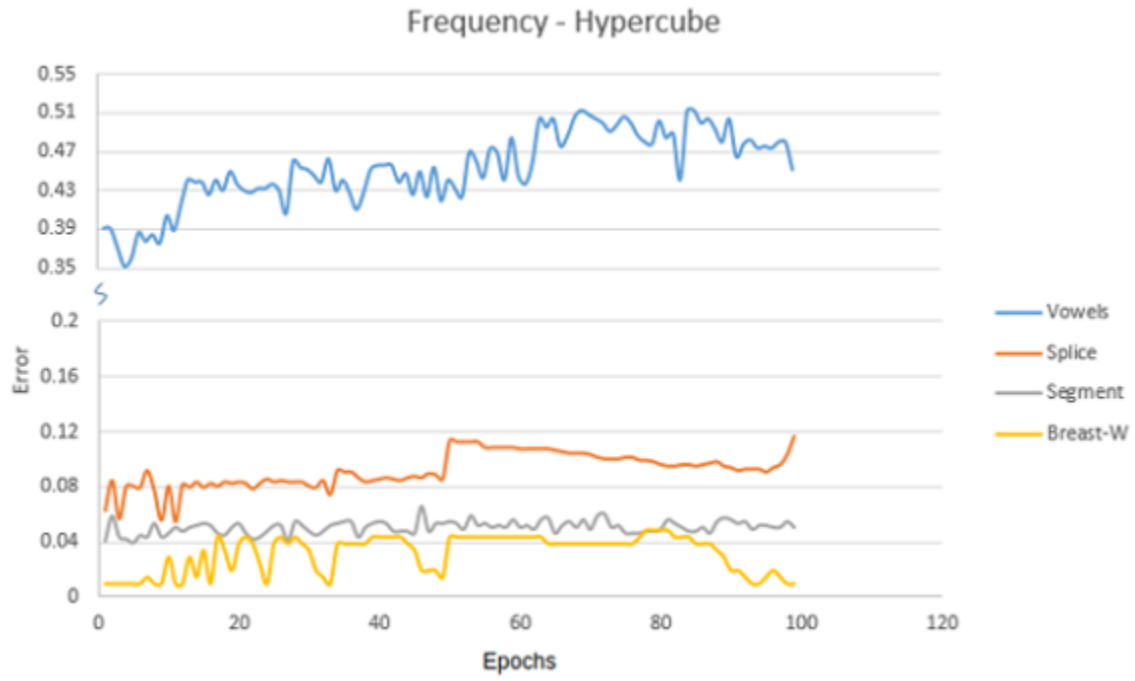

Figure 4.9: Exploring frequency using a Hypercube fusion topology
Using the hypercube fusion topology compares percent error as the frequency increases for the four dataset: vowel, segment, breast-w, splice.

Chapter 5

Conclusion

We have presented an algorithm that enables neural networks, with only a subset of the data, to learn from each other to learn from each other while still achieving levels of accuracy comparable, or better, to an individual network trained on all available data We conducted four experiments that demonstrated that this algorithm achieves this goal. In three experiments Model Fusion made predictions within one percent accuracy of the individual model and in the fourth experiment it made more accurate prediction. This proves our hypothesis that weighted average model fusion can achieve comparable levels of accuracy by allowing neural networks to teach each other. We explored how the meta-parameters fusion rate and fusion frequency affected the accuracy of Model Fusion. Depending on the fusion topology, the fuse rate had varying affects on the overall accuracy. Using random fusion topology, the best fuse rates varies depending on the problem. The other two fusion topologies show more consistency across problems, having similar fuse rates produce the most accurate results in all tests. At a high level higher fusion frequencies proved to make more accurate predictions, but at a more granular level have sporadic effects on the accuracy of Model Fusion.

References

[1] Tim Andersen and Tony Martinez. Wagging: A learning approach which allows single layer perceptrons to outperform more complex learning algorithms. In Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'99, 1999.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010, pages 177–186. Springer, 2010.

[3] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 535–541, New York, NY, USA, 2006. ACM.

[4] Richard Byrne and Andrew Whiten. Machiavellian intelligence: social expertise and the evolution of intellect in monkeys, apes, and humans (oxford science publications). 1989.

[5] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. arXiv preprint arXiv:1511.05641, 2015.

[6] Sung-Bae Cho and Jin H Kim. Multiple network fusion using fuzzy logic. IEEE Transactions on Neural Networks, 6(2):497–501, 1995.

[7] William James Dally and Brian Patrick Towles. Principles and practices of interconnection networks. Elsevier, 2004.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(Jul):2121–2159, 2011.

[9] Michael Gashler. Waffles: A machine learning toolkit. J. Mach. Learn. Res., 12:2383–2387, July 2011.

[10] Giorgio Giacinto and Fabio Roli. Design of effective neural network ensembles for image classification purposes. Image and Vision Computing, 19(9):699–707, 2001.

[11] L. K. Hansen and P. Salamon. Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell., 12(10):993–1001, October 1990.

[12] Esther Herrmann, Josep Call, María Victoria Hernández-Lloreda, Brian Hare, and Michael Tomasello. Humans have evolved specialized skills of social cognition: The cultural intelligence hypothesis. science, 317(5843):1360–1366, 2007.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[14] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. Neural networks, 4(2):251–257, 1991.

[15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[16] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation and active learning. In Proceedings of the 7th International Conference on Neural Information Processing Systems, NIPS'94, pages 231–238, Cambridge, MA, USA, 1994. MIT Press.

[17] S. Lee, S. Purushwalkam, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra. Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. ArXiv e-prints, 2016.

[18] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In Advances in Neural Information Processing Systems, pages 693–701, 2011.

[19] Youcef Saad and Martin H Schultz. Topological properties of hypercubes. IEEE Transactions on computers, 37(7):867–872, 1988.

[20] Christopher Smith and Yaochu Jin. Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction. Neurocomputing, 143:302–311, 2014.

[21] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4(2), 2012.

[22] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In Advances in Neural Information Processing Systems, pages 685–693, 2015.

[23] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. Artif. Intell., 137(1-2):239–263, May 2002.