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

Joshua Smith
Department Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701
Email: jrs023@email.uark.edu

Michael S. Gashler
Department Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701
Email: mgashler@uark.edu

***Abstract*—We investigate a method for cooperative learning called weighted average model fusion that enables neural networks to learn from the experiences of other networks, as well as from their own experiences. Modern machine learning methods have focused predominantly on learning from direct training, but many situations exist where the data cannot be aggregated, rendering direct learning impossible. However, we show that the simple approach of averaging weights with peer neural networks at periodic intervals enables neural networks to learn from second hand experiences. We analyze the effects that several meta-parameters have on model fusion to provide deeper insights into how they affect cooperative learning in a variety of scenarios.**

I. Introduction

The "ultra-social" nature of humans has been recognized as a primary factor in their general cognitive development [12]. However, most machine learning methods still focus on learning exclusively from direct observations. Can machines also learn to benefit from the experiences of others, as humans do so effectively? In this paper, we show that the very simple approach of averaging weights with peer neural networks at periodic intervals is sufficient to facilitate the effective transfer of learned knowledge. Our neural networks are able to demonstrate capabilities they were never directly trained to have, and converge to approximately the same accuracy as a single model that was trained with all of the data.

Iterative weight training methods, such as stochastic gradient descent [3], AdaGrad [8], RMSprop [22], and Adam [15], have been highly effective at refining models to fit training data. These methods all iteratively refine the weights of a model until it fits to some training data. Our approach is based the idea that by periodically combining the weights of two peer networks during this iterative training process, learned knowledge will eventually diffuse throughout a network of peer neural networks.

One application where cooperative learning can make a large impact is mining data from hospitals. There are over five thousand hospitals in the United States alone, and many more throughout the world. Each one has a large storage of patient data. In aggregate, this data presumably contains valuable knowledge that could be mined to discover new patterns for diagnoses or new treatments [1], [18], [14]. However, due to the personal and sensitive nature of this data, hospitals are neither willing nor often even permitted to share this data. Another example where the aggregation of data is not practical is smart grid technology. So many devices can now collect so much data so rapidly that it is not even feasible to transport it all to a central location. Under such conditions, our method proposes that learning can be peformed locally, without ever even aggregating the data. This approach is compelling because learned models can be orders of magnitude smaller than the raw training data, and we show that individual models can effectively learn capabilities for which they were never directly trained.

II. Related Works

Ensemble learning may be the most well-established approach for combining models together. Ensembles of neural networks started with Hansen and Salamon's work [11]. Since that time, there have been numerous studies done that build on their work, such as Multiple Networks Fusion using Fuzzy Logic [6], Ensembling neural networks: Many could be better than all [24], 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 [21] and Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles [17].

Ensemble techniques combine the output or predictions of each neural network in a variety of different ways. This requires significant computational cost because all of the models must first be evaluated to generate the prediction. By contrast, our method reduces computational cost by combining the learning first, and making predictions later. Ensemble techniques are also not sufficient to address applications where

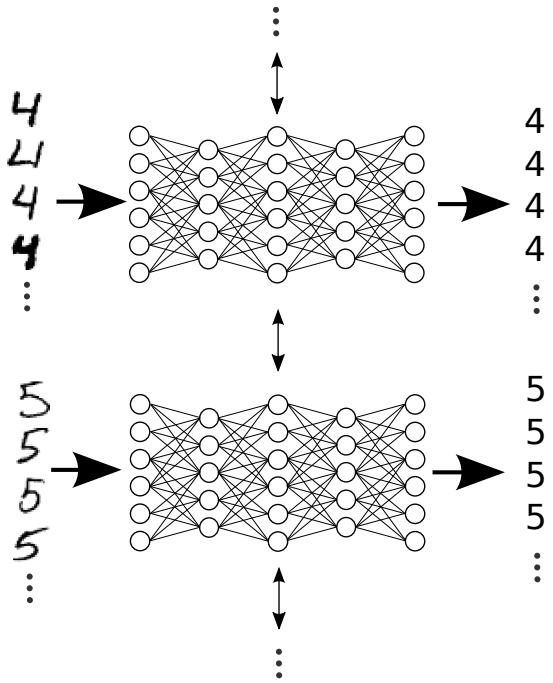

Fig. 1. To demonstrate learning from second-hand experiences, 10 neural networks were trained using the MNIST dataset, such that each neural network was only shown one class of digits. Large arrows represent the raw training data, which does *not* need to be aggregated at a central location for this mode of training. Small bi-directional arrows represent the models (which are typically orders of magnitude smaller than the raw training data) that were exchanged between peers during training.

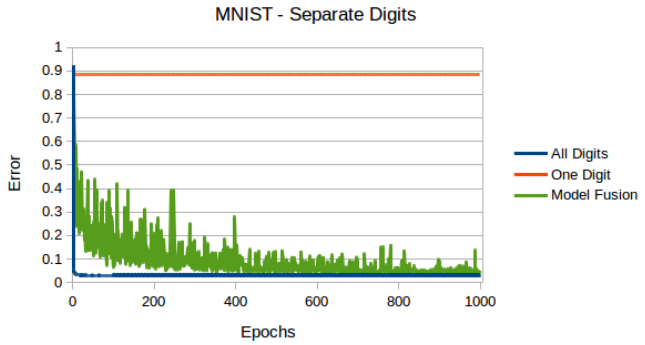

Fig. 2. Comparison of a model trained on all available data (**blue**), a model trained in isolation on just one class of digits (**red**), and a model trained on just one class of digits but also allowed to learn from the second-hand experiences of peer neural networks (**green**). Second-hand learning is somewhat slower than learning from direct experience, but eventually converges to be as accurate as as the model trained on all of the data.

data cannot be aggregated because they generally derive their accuracy from the assumption that all of the models have been trained to address the same problem. Our model, by contrast, can aggregate multiple different capabilities into a single model.

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 [4]. 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.

Another related approach is transfer learning [**?**] [**?**]. Transfer learning methods seek to learn in one domain, then leverage that learning to improve learning in another domain. This seeks to enable the newly created network to apply the knowledge from its predecessor toward learning new tasks. The difference is that transfer learning would be evaluated based on the improvement to learning only in the new domain, whereas cooperative learning would be evaluated based on the ability to exhibit learning from both sources. Another closely related approach is Net2Net [5], which uses an existing teacher network to train a student network. Unlike other algorithms that compress the model [13] [4] [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.

Weight averaging has been used to parallelize stochastic gradient descent. An algorithm called HogWild [19] 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 [23] finds a running average of parallel networks as an update rule. These methods are similar to our approach, but their objective is different. Whereas they are attempting to obtain performance gains, we are seeking a new mode of learning that can operate in cases where data aggregation is prevented, and where models can benefit from potentially dissimilar experiences of peer models in a network.

There has also been work done with averaging the weights of single-layer perceptrons [2], referred to as “Wagging”. “Wagging” seeks to improve predictive accuracy for a single layer perceptron. We seek to enable applications that require training of multilayer perceptrons.

III. Methods

When training a neural network, an optimization method is applied to iteratively refine the weights of the model.

A. Model Fusion

In order two fuse two networks, we combine the weights by

Algorithm 1 Weight Averaging Model Fusion

```
function Model Fusion(N1, N2, F)
  A = weights of N1
  B = weights of N2
  let W be the size of A (which also the size of B).
  for w ∈ W do
    if Aw == 0 then
      Aw = Bw
    else if Bw == 0 then
      Bw = Aw
    else
      Aw = (1.0 - F)*Aw + F*Bw
      Bw = (1.0 - F)*Bw + F*Aw
    end if
  end for
  weights of N1 = A
  weights of N2 = B
end function
```

calculating a pairwise weighted average of their corresponding weights, as described in Algorithm 1. Since each weight needs a corresponding weight, both networks need the same architecture. A "fusion rate" meta-parameter, F, balances the extent to which each model seeks to retain its existing knowlege with the extent to which it is willing to be influenced by the peer model. Intuitively, this parameter directs the rate at which information will flow throughout the network of neural networks. The value for F ranges from 0 to 1. Values less than 0.5 cause the neural networks to favor the knowledge they learns from direct pattern presentations. Values greater than 0.5 cause them to prefer knowledge gained from other neural networks.

As a special case, non-zero weights always dominate over corresponding weights with a value of zero, no matter what value is used for the fusion rate. This case enables model fusion to be used jointly with L1 regularization to promote better utilization of the weights. Since L1 regularization drives weights toward zero, this causes it to make room for more knowledge to be represented.

A "fusion frequency" parameter defines the interval at which the fusion algorithm is applied during the training process.

B. 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.

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

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 ∈ {0, ..., S} do
    if s+1 <S then
      ModelFusion(Ns, Ns+1, F)
    else
      ModelFusion(NS, N0, F)
    end if
  end for
end function
```

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.

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 R1 and R2 be uniformly distributed between 0 and S.
  while S ≥ 2 do
    r1 is a random value between 0 and S.
    r2 is a random value between 0 and S.
    ModelFusion(Nr1, Nr2, 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. In the case where there is an odd number of networks, one network will be randomly fused with two other networks. All 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.

The hypercube topology is based upon a hypercube graph.

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 $x \in \{0, \ldots, n\}$ **do**
    for $s \in \{0, \ldots, S\}$ **do**
      b = binary number of N_s, with the x bit inverted.
      ModelFusion(N_s, N_b, F)
    end for
  end for
end function

Hypercube graphs, also called a n-cube graph, 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 [20] [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.

IV. Results

In this section, we present empirical results that show how weight averaging model fusion affects the transfer of knowledge during iterative learning processes with neural networks. We ran all tests on 5 different datasets: MNIST, CIFAR, Vowel, Image Segmentation, 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.

A. 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.

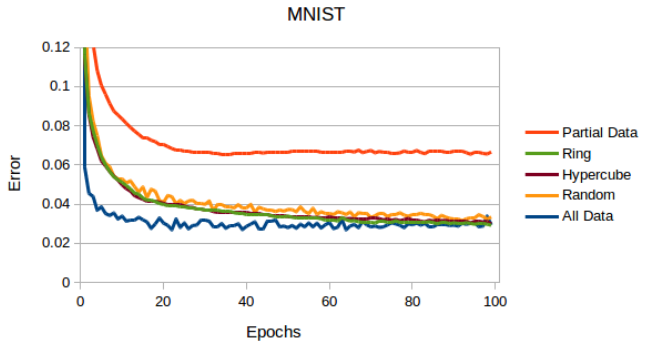

Fig. 3. Comparison of the percent error over time for a network that has trained on all of the data (All Data), one that has trained on part of the data (Partial Data), and the result of different fusion topology techniques Random, Ring an Hypercube. All of the networks in this problem had a learning rate of 0.01 , two hidden layers of 80 and 30 nodes, fusion rate of .55, and a fusion frequency of one epoch.

TABLE I
TABLE OF PREDICTIONS

	Percent Error				
Dataset	Min	Max	Ring	Random	Hyper-cube
MNIST	0.0294	0.0666	0.0292	0.0334	0.0299
CIFAR	0.691131	0.80082	0.718728	0.706829	0.69753
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

We use these two neural networks to create ideal performance bounds for our 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 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 results demonstrate that while learning from the experience of other neural networks is slower than learning from direct experiences, it still achieves comparable levels of accuracy.

The second test we created a collection of 8 neural networks, where each network was 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.

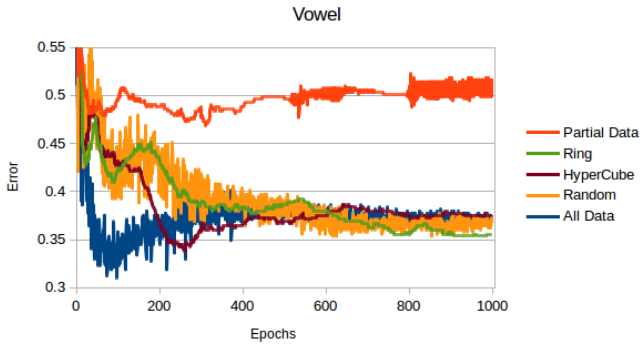

Fig. 4. Comparison of the percent error over time for a network that has trained on all of the data (All Data), one that has trained on part of the data (Partial Data), and the result of different fusion topology techniques Random, Ring an Hypercube. All of the networks in this problem had a learning 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.

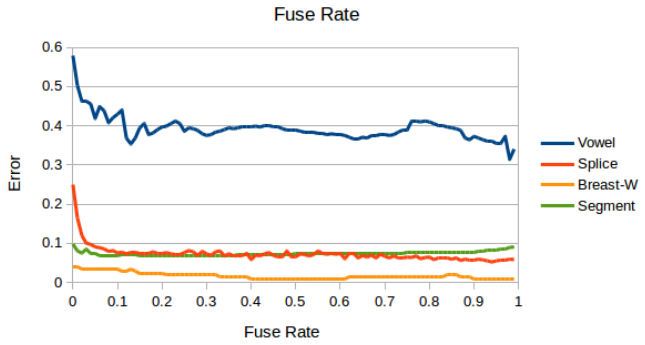

Fig. 5. A plot of error measured with several different datasets after 1000 training epochs while applying fusion at a constant interval, but using varying values for the fusion rate meta parameter. These results exhibit a surprising robustness to the fusion rate parameter, indicating that almost any non-zero value will lead to effective transfer of knowledge between neural networks. Results with ring and hypercube topologies were also obtained. Those exhibit similar trends to those shown in this representative chart.

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 cifar we used 3 hidden layers 3000, 1000 and 10 nodes each, 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. In all 5 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 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 within one percent of the network trained on all of the data.

B. 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. These two meta-parameters may be dependent upon each other; so in order to remove that bias we optimized both of them. We then used the optimal value for the fusion rate in the frequency experiment and the optimal frequency value in the fusion rate experiment.

1) 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 0.00 to 0.99, with a step size of 0.01.

For the experiments using the random fusion topology, the error remained consistent across a wide range of fusion rate above. In the experiments using ring fusion topology and hypercube fusion topology a similar trend held true. These results demonstrates model fusion's robustness to the fusion rate parameter, which indicates that effective transfer of knowledge between neural networks is not heavily dependent upon the weight given to each network.

2) Fusion Frequency: In order to discover the effects that fusion frequency 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

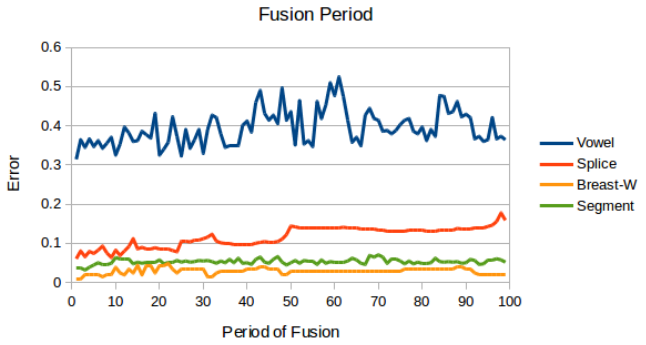

Fig. 6. A plot of error measured with several different datasets after 1000 training epochs while applying fusion at varying intervals. As might be intuitively expected, more frequent fusion generally results in better accuracy with all of the datasets. These results were obtained using the random fusion topology. Results with ring and hypercube topologies were also obtained. Those exhibit similar trends to those shown in this representative chart.

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.

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.

V. Conclusions

We presented an algorithm that enables neural networks, with only a subset of training data, to learn from each other while still achieving levels of accuracy comparable to, or better than, an individual network trained on all available data. We conducted five experiments and in all five our approach made predictions within one percent accuracy of the individual model trained on all of the data. This demonstrates 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. For all fusion topologies model fusion exhibited robustness to the fusion parameter, learning effectively for almost all non-zero values. Higher fusion frequencies made more accurate predictions, but at a more granular level they had sporadic effects on the accuracy of model fusion.

References

[1] E Roland Adams and Anthony Choi. Using neural networks to predict cardiac arrhythmias. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 402–407. IEEE, 2012.

[2] 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.

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

[4] 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.

[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] Bryan Johnson, Alex Bennett, Myungjae Kwak, and Anthony Choi. Automated evaluation of fetal cardiotocograms using neural network. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 408–413. IEEE, 2012.

[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] Rahul Paul, Samuel H Hawkins, Lawrence O Hall, Dmitry B Goldgof, and Robert J Gillies. Combining deep neural network and traditional image features to improve survival prediction accuracy for lung cancer patients from diagnostic ct. In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pages 002570–002575. IEEE, 2016.

[19] 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.

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

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

[22] 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.

[23] 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.

[24] 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.