ARTIFICIAL NEURAL NETWORK SIMPLIFICATION THROUGH ORACLE LEARNING

by

Joshua E. Menke

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

December 2002

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Joshua E. Menke

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Tony R. Martinez, Chair
Date	Dan Ventura
Date	Scott Woodfield

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Joshua E. Menke in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Tony R. Martinez
Chair, Graduate Committee

Accepted for the Department

David W. Embley
Graduate Coordinator

Accepted for the College

G. Rex Bryce,
Associate Dean, College of Physical and Mathematical Science

ABSTRACT

ARTIFICIAL NEURAL NETWORK SIMPLIFICATION THROUGH ORACLE LEARNING

Joshua E. Menke

Department of Computer Science

Master of Science

Often the best model to solve a real world problem is relatively complex. The following thesis presents oracle learning, a method using a larger model as an oracle to train a smaller model on unlabeled data in order to obtain (1) a simpler acceptable model and (2) improved results over standard training methods on a similarly sized smaller model. In particular, this thesis looks at oracle learning as applied to multi-layer perceptrons trained using standard backpropagation. Using multi-layer perceptrons for both the larger and smaller models, oracle learning obtains a 15.16% average decrease in error over direct training while retaining 99.64% of the initial oracle accuracy on automatic spoken digit recognition. For optical character recognition, oracle learning results in an 11.40% average decrease in error over direct training while maintaining 98.95% of the initial oracle accuracy.

ACKNOWLEDGMENTS

I would like to first thank my wife Maren whose constant encouragement and support were second only to a loving Heavenly Father's and without whom I would never have had the stamina to finish this work. Thanks go even to my young daughter Aria who was always willing to break me away from my work to come play with her. I am also very grateful for the counsel given by my advisor, Dr. Tony Martinez, for his constant input, direction, and inspiration in this area. I thank all the members of the Neural Network and Machine Learning research group here in BYU's Computer Science department for always being there for consultation and for not being afraid to tell me if I was wrong. Finally, thanks to all those who have lent encouragement and support including close family and friends.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Machine learning has become a powerful tool for increasing the usefulness of today's computers. As Tom Mitchell states in [1], "A successful understanding of how to make computers learn would open up many new uses of computers and new levels of competence and customization." In fact, as Mitchell continues, "Many practical computer programs have [already] been developed to exhibit useful types of learning, and significant commercial applications have begun to appear." The programs Mitchell refers to are based on algorithms designed to apply learning techniques to automatically find solutions to difficult, real-world problems. Examples of machine learning algorithms include decision trees [2, 3, 19], artificial neural networks [4], bayesian learning [11], nearest neighbor and instance based learning [11], genetic algorithms [15, 16], and many others (see [1] for more). Common applications for machine learning include handwritten character recognition [5, 6], speech recognition [7, 8, 9], face recognition [10], and text classification [12, 13, 14]. "For problems such as speech recognition, algorithms based on machine learning outperform all other approaches that have been attempted to date" [1].

As Le Cun, Denker, and Solla observed in [17], often the best *artificial neural*

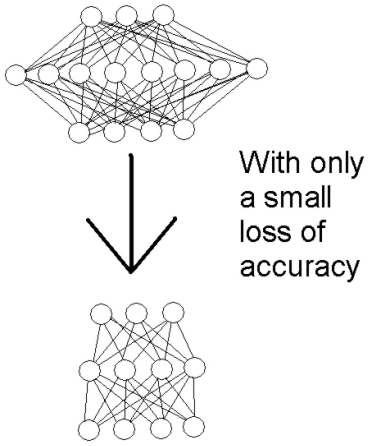

Figure 1.1: Main Goal

network (ANN) to solve a real-world problem is relatively complex. They point to the large ANNs Waibel used for phoneme recognition in [7] and the ANNs of Le Cun et al. with handwritten character recognition in [6]. "As applications become more complex, the networks will presumably become even larger and more structured" [17]. The following research presents the *oracle learning* algorithm, a training method that seeks to accomplish the goal diagrammed in figure 1.1, namely to create less complex ANNs that (1) still maintain an acceptable degree of accuracy, and (2) provide improved results over standard training methods.

Designing a neural network for a given application requires first determining the optimal size for the network in terms of accuracy on a test set, usually by increasing its size until there is no longer a significant decrease in error. Once found, the preferred size for more complex problems is often relatively large. One method of reducing the complexity is to use a smaller ANN still trained using standard methods. Using ANNs smaller than the optimal size results in a decrease in accuracy. The goal of this thesis is to increase the accuracy of these smaller, less resource intensive ANNs

using oracle learning.

As an example consider designing an ANN to recognize spoken digits for dialing on a cellular phone. The network has to be small, fast, and accurate. Now suppose the most accurate digit recognizing ANN given the available training data has 800 hidden nodes, but the resources on the phone allow for only 100 hidden nodes. One solution is to train a 100 hidden node ANN using standard methods, resulting in a compromise of significantly reduced accuracy for a smaller size. This research demonstrates that applying oracle learning to the same problem results in a 100 hidden node ANN that does not suffer from nearly as significant a decrease in accuracy. Oracle learning uses the original 800 hidden node ANN as an oracle to create as much training data as necessary using unlabeled speech data. The oracle labeled data is then used to train a 100 hidden node network to approximate the 800 hidden node network. The results in Chapter 4 show the oracle learning ANN retains 99.9% of the 800 hidden node ANN's accuracy on average, while being $\frac{1}{8}$ the size. The resulting *oracle-trained network* (OTN) is more than 10% more accurate on average than the standard trained 100 hidden node ANN.

Although the previous example deals exclusively with ANNs, oracle learning can be used to train any model using a more accurate model of any type. Both the oracle model and the *oracle-trained model* (OTM) in figure 1.2 can be any of the machine learning models mentioned above (e.g. an ANN, a nearest neighbor model, a bayesian learner, etc.). In fact, the oracle model can be any arbitrary functional mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ where n is the number of inputs to both the mapping and the OTM, and m is the number of outputs from both. As seen in figure 1.2, the same unlabeled data is fed into both the oracle and the OTM, and the error used to train the OTM is the oracle's output minus the OTM's output. Thus the OTM learns to minimize its differences with the oracle on the unlabeled data set. Since

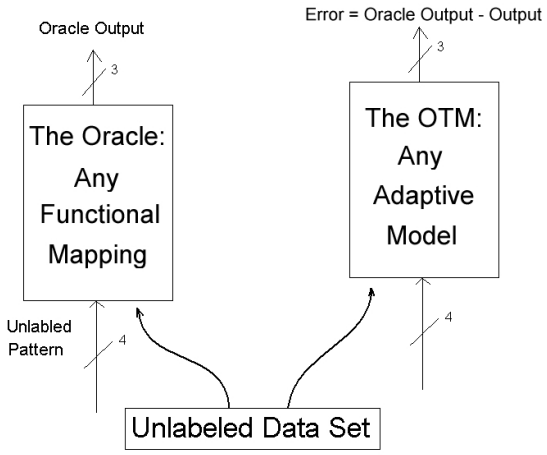

Figure 1.2: Oracle Learning Summary

the following research uses multilayer feed-forward ANNs with a single-hidden layer as both oracles and OTMs, the rest of the paper describes oracle learning in terms of ANNs. An ANN used as an oracle is referred to as an *oracle ANN* (a standard backpropagation trained ANN used as an oracle). The following nomenclature used for referring to OTNs:

$$\text{OTN}(n \rightarrow m)$$

reads "an OTN approximating an n hidden node ANN with an m hidden node ANN." For example:

$$\text{OTN}(800 \rightarrow 100)$$

reads "an OTN approximating an 800 hidden node ANN with a 100 hidden node ANN."

The idea of approximating a more complex model is not new. Domingos used Quinlan's C4.5 decision tree approach from [19] in [18] to approximate a bagging ensemble (bagging is a method of combining models, see [21] for details) and Zeng and Martinez used an ANN in [20] to approximate a similar ensemble (both using

the bagging algorithm Breimen proposed in [21]). Craven and Shavlik used a similar approximating method to extract rules [22] and trees [23] from ANNs. Domingos and Craven and Shavlik used their ensembles to generate training data where the targets were represented as either being the correct class or not. Zeng and Martinez used a target vector containing the exact probabilities output by the ensemble for each class. The following research also uses vectored targets similar to Zeng and Martinez since Zeng's results support the hypothesis that vectored targets "capture richer information about the decision making process ..." [20]. While previous research has focused on either extracting information from neural networks [22, 23] or using statistically generated data for training [18, 20], the novel approach presented here is that currently unused, unlabeled data be labeled using the more complex model as an oracle.

Chapter 2

ORACLE LEARNING

Oracle learning consists of the following 3 steps:

1. Obtaining the Oracle
2. Labeling the Data
3. Training the OTN

2.1 Obtaining the Oracle

The primary component in oracle learning is the oracle itself. Since the accuracy of the oracle ANN directly influences the performance of the final, simpler ANN, the oracle must be the most accurate classifier available, regardless of complexity (number of hidden nodes). In the case of ANNs, the most accurate classifier is usually the largest ANN that improves over the next smallest ANN. For example, an 800 hidden node ANN that shows significantly better accuracy than any smaller ANN would be an oracle if no larger ANN is more accurate. The only requirement is that the number and type of the inputs and the outputs of each ANN (the oracle and the OTN) match. For the following experiments, the oracle is found by testing ANNs with increasingly

more hidden nodes until there is no longer a significant increase in accuracy and then choosing the size that demonstrates both a high mean and a low variance.

Notice that by definition of how the oracle ANN is chosen, any smaller, standard-trained ANN must have a significantly lower accuracy. This means that if a smaller OTN approximates the oracle such that their differences in accuracy become insignificant, the OTN will have a higher accuracy than any standard-trained ANN of its same size—regardless of the quality of the oracle.

2.2 Labeling the Data

The main step in oracle learning is to use the oracle ANN to create a very large training set for the OTN to use. Fortunately the training set does not have to be pre-labeled since the OTN only needs the oracle ANN's outputs for a given input. Therefore the training set can consist of as many data points as there are available, including unlabeled points.

The key to the success of oracle learning is to obtain as much data as possible that ideally fits the distribution of the problem. There are several ways to approach this. In [20], Zeng and Martinez use the statistical distribution of the training set to create data. However, the complexity of many applications makes accurate statistical data creation very difficult since the amount of data needed increases exponentially with the dimensionality of the input space. Another approach is to add random jitter to the training set according to some (a Gaussian) distribution. However, early experiments with the jitter approach did not yield promising results. The easiest way to fit the distribution is to have more real data. In many problems, like *automatic speech recognition* (ASR), there are more than enough unlabeled real data that can be used for oracle learning. Other problems where there are an abundance of unlabeled data include intelligent web document classifying, optical character recognition, and any other problem where gathering the data is far easier than labeling them. The

oracle ANN can label as much of the data as necessary to train the OTN and therefore the OTN has access to an arbitrary amount of training data distributed as they are in the real world.

To label the data, this step creates a target vector $\boldsymbol{t}^j = t_1 \ldots t_n$ for each input vector $\boldsymbol{x}^j$ where each t_i is equal to the oracle ANN's activation of output i given the j^{th} pattern in the data set, $\boldsymbol{x}^j$. Then, the final oracle learning data point contains both $\boldsymbol{x}^j$ and $\boldsymbol{t}^j$. In order to create the labeled training points, each available pattern $\boldsymbol{x}^j$ is presented as a pattern to the oracle ANN which then returns the output vector $\boldsymbol{t}^j$. The final oracle learning training set then consists of the pairs $\boldsymbol{x}^1\boldsymbol{t}^1 \ldots \boldsymbol{x}^m\boldsymbol{t}^m$ for all m of the previously unlabeled data points.

Once again, Zeng and Martinez found the use of vectored targets to give improved accuracy over using standard targets in [20].

2.3 Training the OTN

For the final step, the OTN is trained using the data generated in step 2, utilizing the targets exactly as presented in the target vector. The OTN interprets each real-valued element of the target vector $\boldsymbol{t}^j$ as the correct output activation for the output node it represents given $\boldsymbol{x}^j$. The backpropagated error is therefore $t_i - o_i$ where t_i is the i^{th} element of the target vector $\boldsymbol{t}^j$ (and also the i^{th} output of the oracle ANN) and o_i is the output of node i. This error signal causes the outputs of the OTN to approach the target vectors of the oracle ANN on each data point as training continues.

As an example, the following vector represents the output vector $\boldsymbol{o}$ for the given input vector $\boldsymbol{x}$ of an oracle ANN. Notice the 4^{th} output is the highest and therefore the correct one as far as the oracle ANN is concerned.

$$\langle 0.27, 0.34, 0.45, 0.89, 0.29 \rangle \tag{2.1}$$

Now suppose the OTN outputs the following vector:

$$\langle 0.19, 0.43, 0.3, 0.77, 0.04 \rangle \tag{2.2}$$

The oracle-trained error is the difference between the target vector in 2.1 and the output in 2.2:

$$\langle 0.08, -0.09, 0.15, 0.12, 0.25 \rangle \tag{2.3}$$

In effect, using the oracle ANNs outputs as targets for the OTNs makes the OTNs real-valued function approximators learning to behave like their oracles.

The size of the OTN network is chosen according to the given resources. If a given application calls for ANNs no larger than 20 hidden nodes, then a 20 hidden node OTN is created. If there is room for a 200 hidden node network, then 200 hidden nodes is preferable. If the oracle itself meets the performance constraints, then, of course, it should be used in place of an OTN.

Chapter 3

METHODS

The following experiments serve to validate the effectiveness of oracle learning, demonstrating the conditions under which oracle learning best accomplishes its goals. Trends for increasing the relative amount of oracle-labeled data are shown by repeating each experiment using smaller amounts of hand-labeled data while keeping the amount of unlabeled data constant. Experiments to determine the effects of removing or adding to the unlabeled data set while keeping the amount of hand-labeled data constant will be conducted as subsequent research because of the amount of time required to do them.

3.1 Automatic Speech Recognition

3.1.1 The Application

One of the most popular applications for smaller computing devices (i.e. hand held organizers, cellular phones, etc.) and other embedded devices is ASR. Since the interfaces are limited in smaller devices, being able to recognize speech allows the user to more efficiently enter data. Given the demand for and usefulness of speech recognition in systems lacking in memory and processing power, there is a need for simpler ASR engines capable of achieving acceptable accuracy.

The ASR engine used for the experiment is shown in figure 3.1. Inputs are fed into an ANN and ANN phoneme outputs are fed into a decoder that builds the phonemes into words. Any ANN can be used as the neural network recognizer part of the engine when determining word and utterance accuracy. The following experiment reduces the complexity of that ANN.

3.1.2 The Data

The following experiment uses data from the unlabeled TIDIGITS corpus [27] for testing the ability of the oracle ANN to create accurate phoneme level labels for the OTN. The inputs are the first 13 Mel cepstral coefficients and their derivatives in 16 ms intervals extracted in 10 ms overlapping frames. The TIDIGITS corpus is partitioned into a training set of 15,322 utterances (around 2,700,000 phonemes), a hold-out set of 1,000 utterances, and a test set of 1,000 utterances (both 180,299 phonemes). Four subsets of the training corpus consisting of 150 utterances (26,000 phonemes), 500 utterances (87,500 phonemes), 1,000 utterances (175,000 phonemes), and 4,000 utterances (700,000 phonemes) are *bootstrap*-labeled at the phoneme level and used for training the oracle ANN. Only a small amount of speech data has ever been phonetically labeled because it is inaccurate and expensive. Bootstrapping involves using a trained ANN to force align phoneme boundaries given word labels to create $0-1$ target vectors at the phoneme level. Forced alignment is the process of automatically assigning where the phonemes begin and end using the known word labels and a trained ANN to estimate where the phonemes break and what the phonemes are. Although the bootstrapped phoneme labels are only an approximation, oracle learning succeeds as long as it can effectively reproduce that approximation in the OTNs. Each experiment is repeated using each of the above subsets as the only available labeled data in order to determine how varying amounts of unlabeled data affect the performance of OTNs.

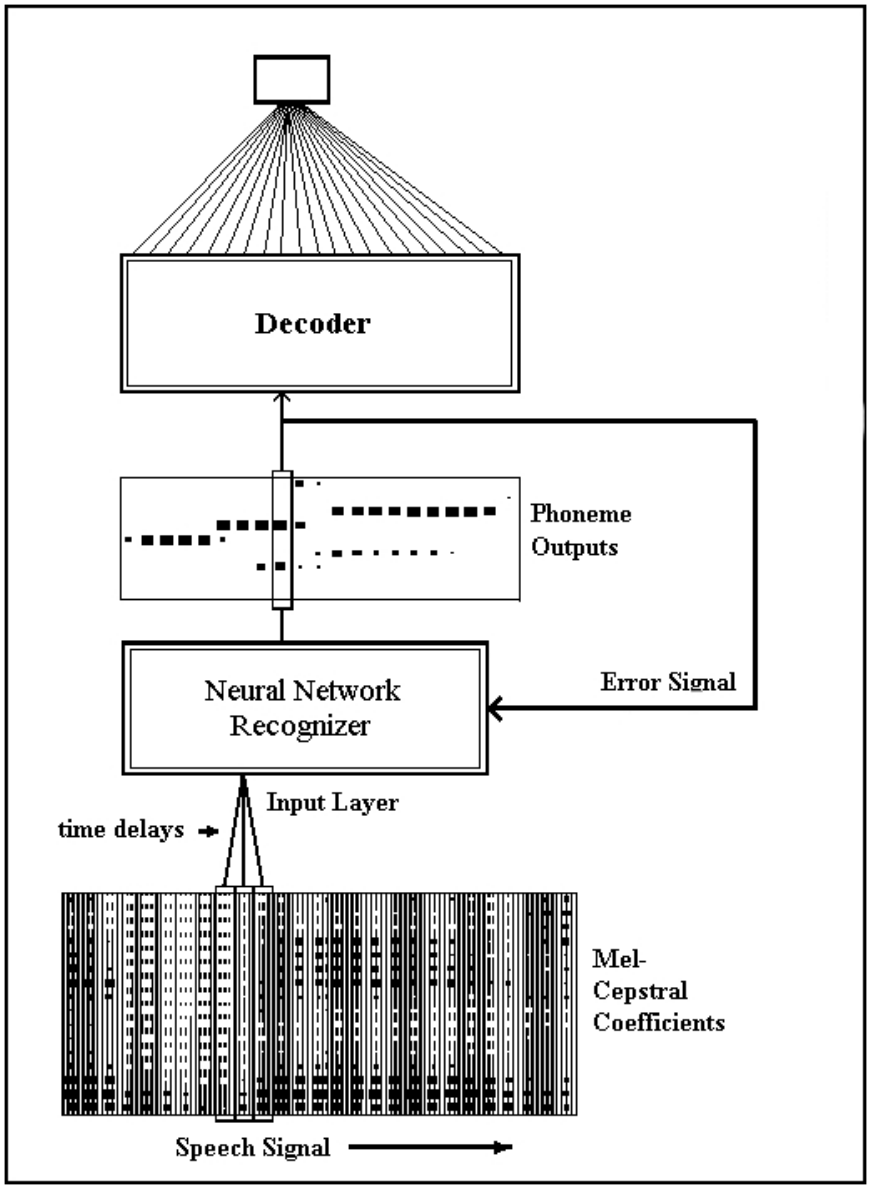

Figure 3.1: The basic ASR Engine

Two important questions that need addressing given the above training sets are (1) whether it is always better to use bootstrapping instead of oracle learning, and (2) whether any of the above training sets contain sufficient data to learn a good solution. The answer to (1) is not relevant in this research since bootstrapping still requires data hand-labeled at the word level. Oracle learning can use data without any hand-labeling to produce theoretically larger training sets than bootstrapping can. The answer to (2) is yes in this case since prior experience with the TIDIGIT training set shows that training on all 15,322 utterances does not lead to significant improvement over training on only 4,000 utterances.

3.1.3 Obtaining the Oracles

For each training set (of the four sizes listed in 3.1.2), ANNs of an increasing number of hidden nodes are trained and tested on the hold-out set. The size of the oracle ANN is chosen as the ANN with the highest average and least varying word accuracy (averaged over five different random initial weight settings). The oracle selection process is repeated for each training set, resulting in an oracle chosen for each of the four training set sizes. Figures 3.2-3.5 graph the mean accuracy and standard deviation on the hold-out set of the most accurate ANNs by number of hidden nodes for each training set size. Except for the 1% training set size case, the ideal oracle ANN size is 800 hidden nodes since 800 hidden node ANNs consistently have the highest mean and lowest standard deviation. In the 1% case, the best oracle is the 250 hidden node ANN because although its mean is slightly lower than the 200 hidden node case, its standard deviation is lower, making it more likely to behave similarly on the test set. Therefore, for the three largest training sets, the best oracles have 800 hidden nodes, and for the smallest training set, 250 hidden nodes.

The same decaying learning rate is used to train every ANN (including the OTNs) and starts at 0.025, decaying according to $\frac{.025}{1+\frac{p}{5N}}$ where p is the total number of patterns

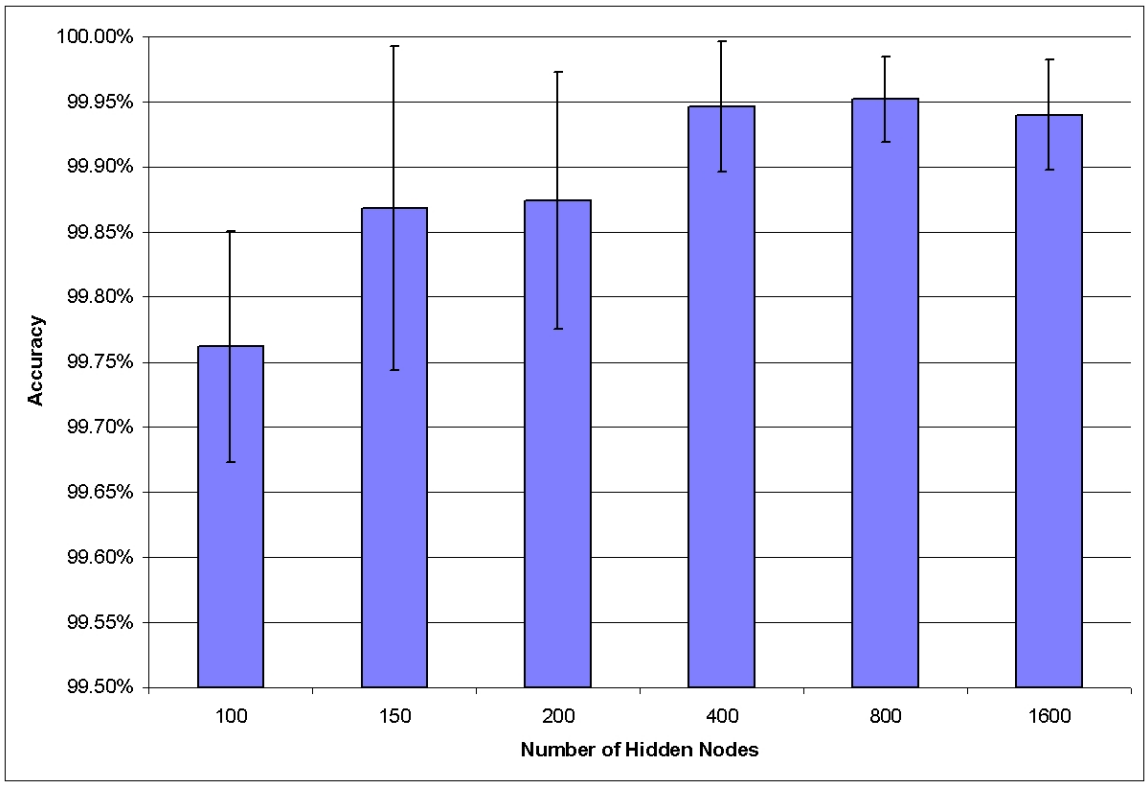

Figure 3.2: Mean accuracy with standard deviation error bars for $100 - 1600$ hidden node ANNs on the 4,000-utterance training set. The 800 node net has the highest accuracy and the lowest standard deviation.

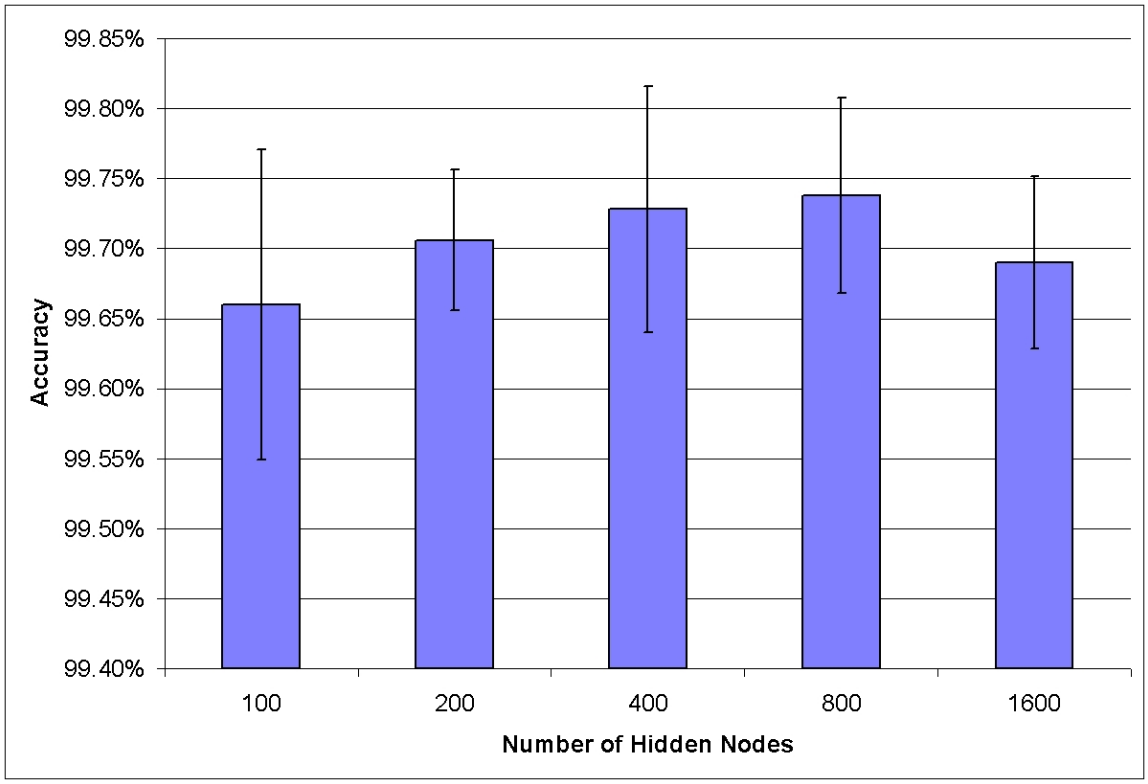

Figure 3.3: Mean accuracy with standard deviation error bars for $100 - 1600$ hidden node ANNs on 1,000-utterance training set. The 800 node net is once again preferred.

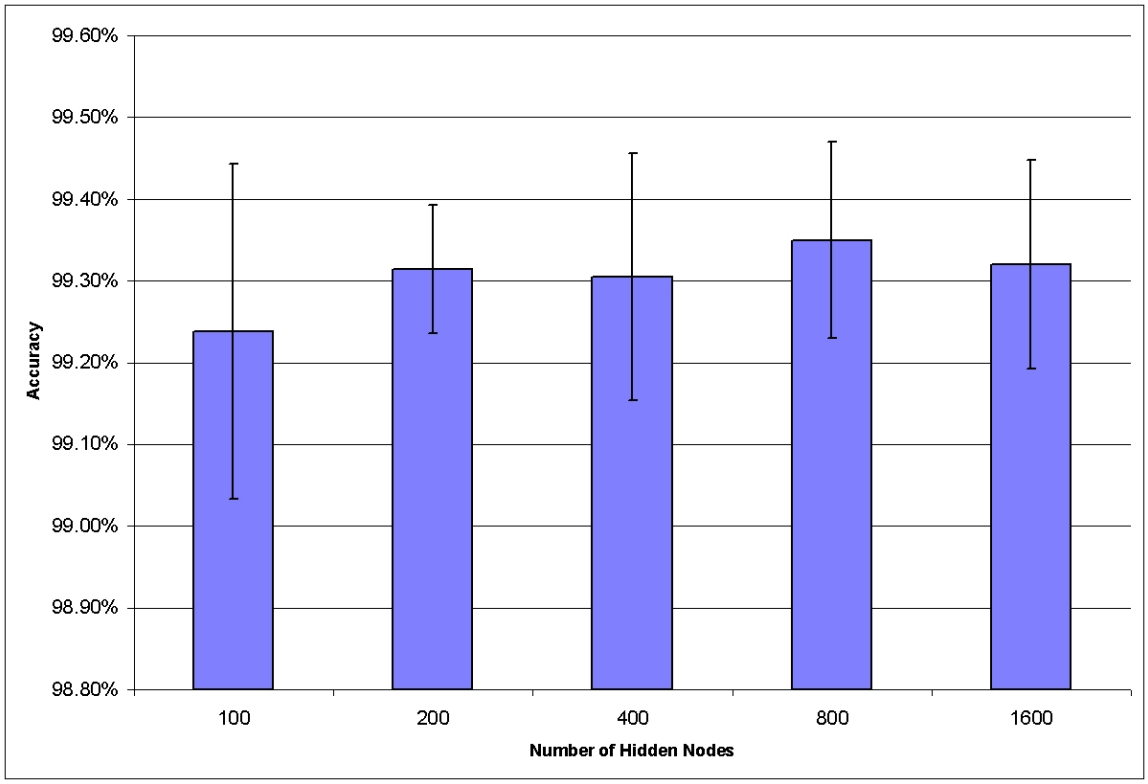

Figure 3.4: Mean accuracy with standard deviation error bars for $100 - 1600$ hidden node ANNs on the 500-utterance training set with similar results to the preceding figures.

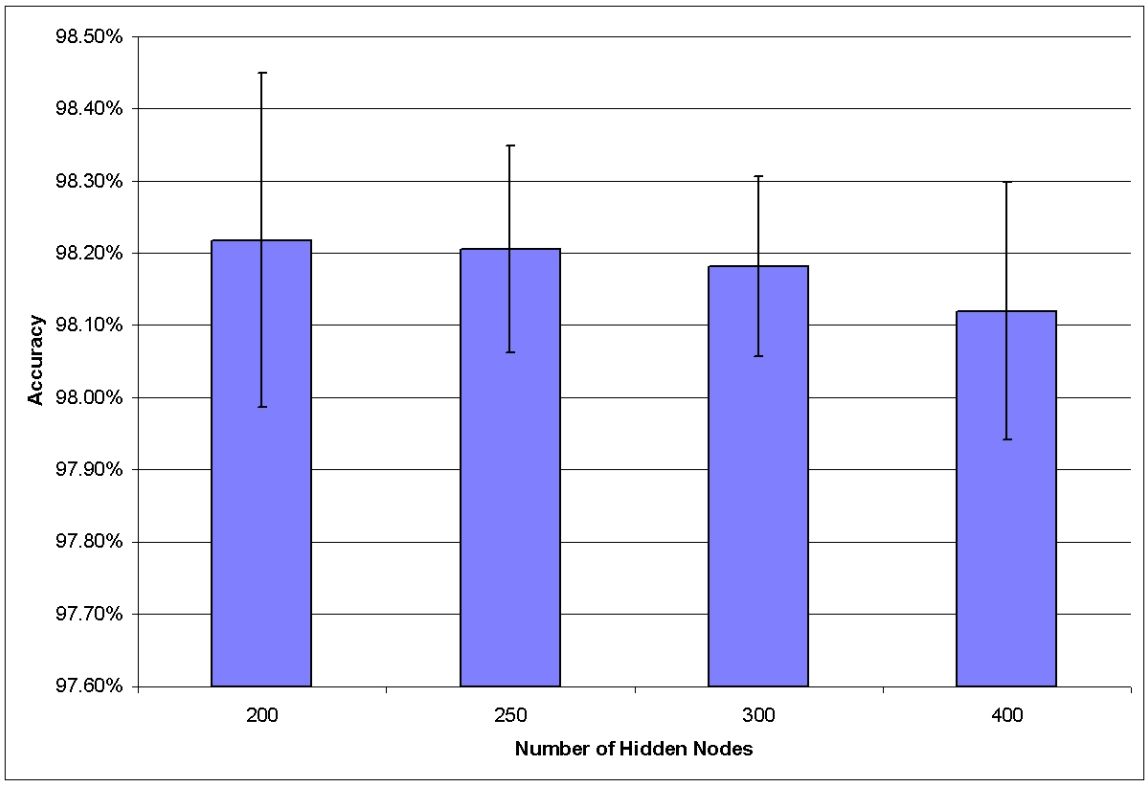

Figure 3.5: Mean accuracy with standard deviation error bars for $200 - 400$ hidden node ANNs using only 150 utterances. The 200 hidden node ANN has a slightly higher mean accuracy but higher standard deviation than the chosen 250 hidden node ANN.

seen so far and N is the number of patterns in the training set. This has the effect of decaying the learning rate by $\frac{1}{2}$ after five epochs, $\frac{1}{3}$ after 10, $\frac{1}{4}$ after 15, etc. The learning rate is chosen for its favorable performance in past experiments.

3.1.4 Labeling the Data Set

For the next step a large training set is created from the unlabeled data for each of the training set sizes. The entire $15,000+$ utterance set of unlabeled data (not including the 1,000 utterance hold-out and test sets) is used to create a new training set consisting of the inputs from the old set combined with the target vectors from the oracle corresponding to that training set size (see 3.1.3), acquiring the target vectors as explained in 2.2 (from the oracle ANN's outputs). Since there are four different oracles (one for each training set size) four separate oracle training sets are created. Oracle learning presents the oracle with an input pattern and then saves the activations of each output node for that input as a vector. The new OTN's training vector then consists of the original input and the new target vector.

3.1.5 Training the OTNs

Finally, each of the large OTN training sets from the previous step are used to train ANNs using the oracle's exact outputs for targets. For a given training pattern, the back-propagated error is $t_i - o_i$ where t_i is the oracle's output for class i and o_i is the output of the OTN on class i.

Since there are no true constraints on the size of the OTN for the experiment (since desktop workstations are used, not smaller devices), the OTN sizes are chosen as fractions of the oracle's size that do not already attain comparable accuracy, specifically 100, 50, and 20 hidden node ANNs. OTNs of each of the above sizes are trained and their accuracies are compared to the ANNs of equivalent size trained without oracles (see Chapter 4).

3.1.6 Performance Criteria

For every training set size in 3.1.2, and for every OTN size, five separate OTNs are trained using the training sets described in 3.1.4, each with different, random initial weights. Even though there are only five experiments per OTN size on each training set, there are a total of 20 experiments for each of the three sizes across the four training sets and 15 experiments for each of the four training sets across the three sizes (for a total of $20 \cdot 3$ or $15 \cdot 4 = 60$ experiments). Therefore, section 4.1 includes results that average across the training sets for a given OTN size, and results that average across the different sizes for a given training set. An overall average is also reported.

After every oracle learning epoch, word accuracies are gathered using the hold-out set, and the respective OTN weights saved. Training continues until there is no improvement in word accuracy on the hold-out set for 30 epochs. The ANN most accurate on the hold-out set is then tested on the test set for a less biased measure of the OTN's performance. The test set results from the five OTNs performing best on the hold-out set are then averaged for the final performance metric.

While word accuracy is perhaps more practically significant, the ANNs for this experiments are trained directly on phonemes and not on words. Training on words requires an ANN output for every word the ASR engine recognizes and although this particular experiment only recognizes digits (requiring 10 outputs), the ASR engine used is designed for general purpose vocabularies requiring the recognition of tens of thousands of words. An ANN trained to recognize tens of thousands of words would be too large to train in a reasonable amount of time. Also, breaking words down into smaller phoneme segments allows for a finer and more consistent granularity in the feature extraction process. It is easier to extract a 16 ms frame every 10 ms than it is to try and determine the length of the current word to extract an appropriately sized

frame. Therefore, since the ANNs are trained on phonemes and not words, the OTNs train to approximate their oracles' phoneme classification behaviors, not their word classification behaviors. Word accuracy is obtained through the use of the phoneme to word decoder shown in figure 3.1. The decoder adds a level of indirection to the training and testing procedure but as long as the OTNs learn to produce output vectors similar enough to the their oracles' output vectors, accuracies after decoding are still comparable to the oracle ANNs' scores. OTNs have theoretically high word scores as long as their output vector distribution is similar to their oracles'. The experiment in the next section (3.2) does not deal with a decoder and so the results are more directly observable.

3.2 Optical Character Recognition

3.2.1 The Application

One of the problems with the ASR application is the element of indirection the decoder adds to determining accuracy. It is possible oracle learning does well on problems with a decoder and struggles on those without. Therefore, a non-decoded experiment is also conducted.

A popular application for ANNs is *optical character recognition* (OCR) where ANNs are used to convert images of typed or handwritten characters into electronic text. OCR is a complex, real word problem, and good for validating oracle learning. It is also good for proving oracle learning's potential because (1) the data points are labeled at the classification or letter level (no bootstrapping) and (2) no decoder is used. It serves as a second major application to validate oracle learning.

3.2.2 The Data

The OCR data set consists of 500,000 alphanumeric character samples partitioned into a 400,000 character training set, a 50,000 character hold-out set, and a 50,000 character test set. Similar to the ASR application in 3.1, four separate training sets

are created, one using all of the training data (400,000 out of the 500,000 sample set), another using 25% of the training data (100,000 points), the third using 12.5% of the data (50,000 points), and the last using only 5% of the training data (4,000 points). Once again, this is done in order to determine the affect of varying the relative amount of data the OTNs "see" yielding cases where the OTN sees 20, 8, and 4 times more data than the standard trained networks, and even the case where they both see the same amount of data. In every case the 400,000-sample training set is used to train the OTN. Holding out parts of the available training data allows the experiments to demonstrate the effectiveness of oracle learning in situations where there are more unlabeled than labeled data available.

For OCR, the case using all of the data represents a situation where the standard ANNs have sufficient data to learn the problem.

3.2.3 Obtaining the Oracles

The OCR ANNs are of the same type as the ASR ANNs; feed-forward single hidden layer networks trained using standard backpropagation. The same decaying learning rate is used for OCR as for ASR, save that it starts at 0.1 instead of 0.025. The learning rate is based on past experiments with OCR. For testing, the highest ANN output classifies the corresponding character. The ANNs are trained in the same way as in ASR experiment, storing the ANN weight configurations for future testing. To determine the best size ANN (oracle ANN) for each of the four training sets, just as with ASR (see section 3.1.3), ANNs of increasing sizes (starting at 32 hidden nodes and doubling) are trained on each set to find the best oracle ANN. The oracle ANN is chosen as the ANN with the highest mean accuracy and lowest standard deviation averaged over five ANNs. Figures 3.6-3.9 graph the mean accuracy and give error bars representing two standard deviations in accuracy for the 5 ANNs averaged. In figure 3.6, using 100% of the training data, the 2,048 hidden node ANN

is chosen over the 4,096 hidden node ANN because (1) their mean accuracies are very close, and (2) the 2,048 hidden node ANN's standard deviation is smaller than the 4,096 hidden node ANN's and therefore less likely to vary. The case where 25% of the training set is used (see figure 3.7) shows a similar situation occurring between the 1,024 hidden node ANN and the 2,048 hidden node ANN, where the 1,024 hidden node ANN is chosen as the oracle ANN. For the 12.5% case, the 4,096 hidden node and 2,048 hidden ANNs are almost identical in accuracy and the 4,096 hidden node ANN is only chosen since it performs slightly better. An 8,192 hidden node ANN is not trained since the 4,096 hidden node ANN did not improve appreciably over the 2,048 hidden node ANN. Finally, when using only 5% of the training set to train the ANNs (see figure 3.9), the accuracies are once again very close, but the 2,048 hidden node ANN is chosen as the oracle for being slightly better. In all of these cases, the ANNs are usually too close in accuracy to be to say one is definitely better than the other, so the methods used in this section to choose one above the other are in essence only tie-breakers between the best of the ANNs.

3.2.4 Labeling the Data

For the next step a large training set is created by labeling from the entire 400,000 character training set with each of the four oracles chosen in 3.2.3. This creates four new training sets consisting of the inputs from the old set combined with the target vectors from each oracle ANN, acquiring the target vectors from the oracle ANN's outputs as explained in 2.2.

3.2.5 Training the OTNs

The large OTN training sets described in 3.2.4 are used to train ANNs of sizes beginning with the first major break in accuracy, starting at either 512 or 256 hidden nodes and decreasing by halves until 32 hidden nodes.

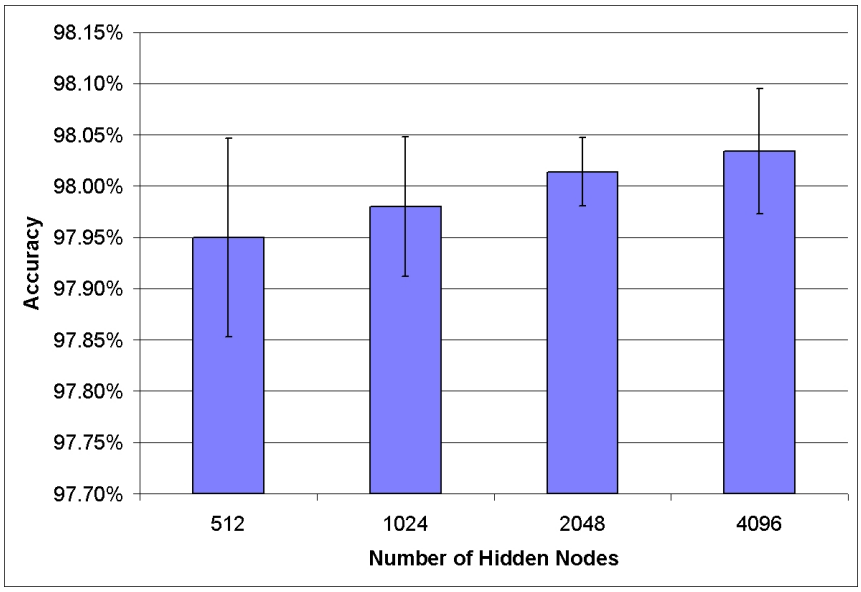

Figure 3.6: Mean accuracy with standard deviation for ANNs using 100% of the training data. The 4096 hidden node ANN has the highest accuracy, but the 2048 hidden node ANN is nearly as accurate and varies less in its accuracy.

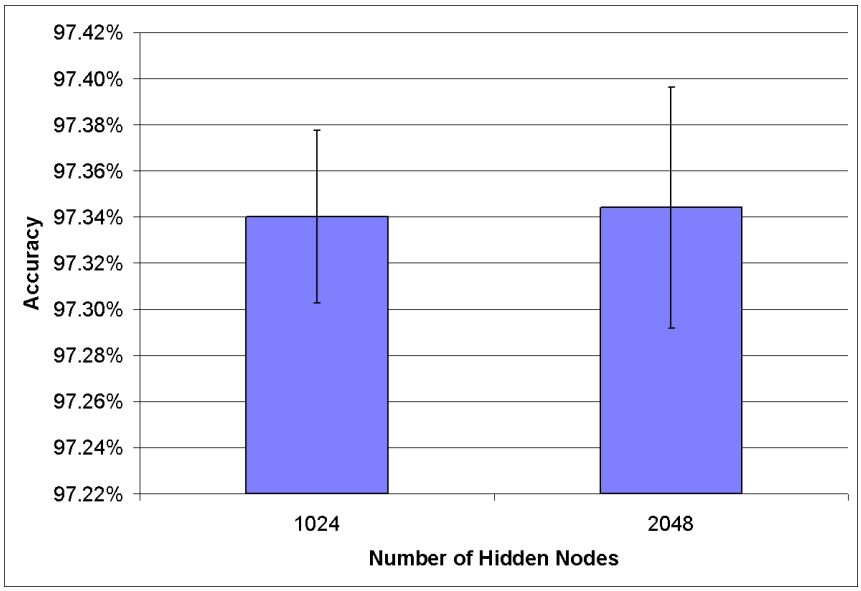

Figure 3.7: Mean accuracy with standard deviation using 25% of the training data. The 1,024 hidden node ANN is the chosen oracle since its accuracy is almost identical to the 2,048 hidden node ANN, but it deviates slightly less from its mean.

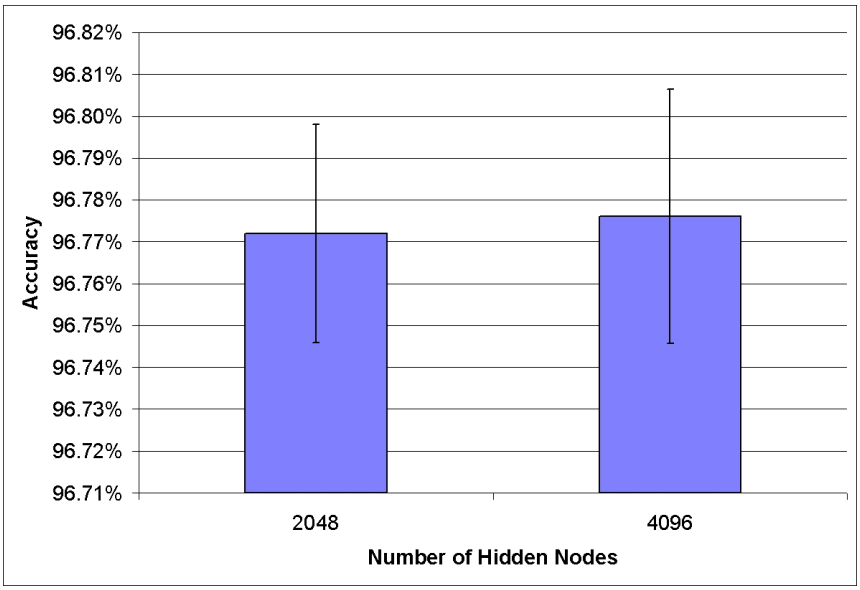

Figure 3.8: Mean accuracy with standard deviation using 12.5% of the training data. Although almost identical, the 4,096 ANN is chosen for its slightly better performance.

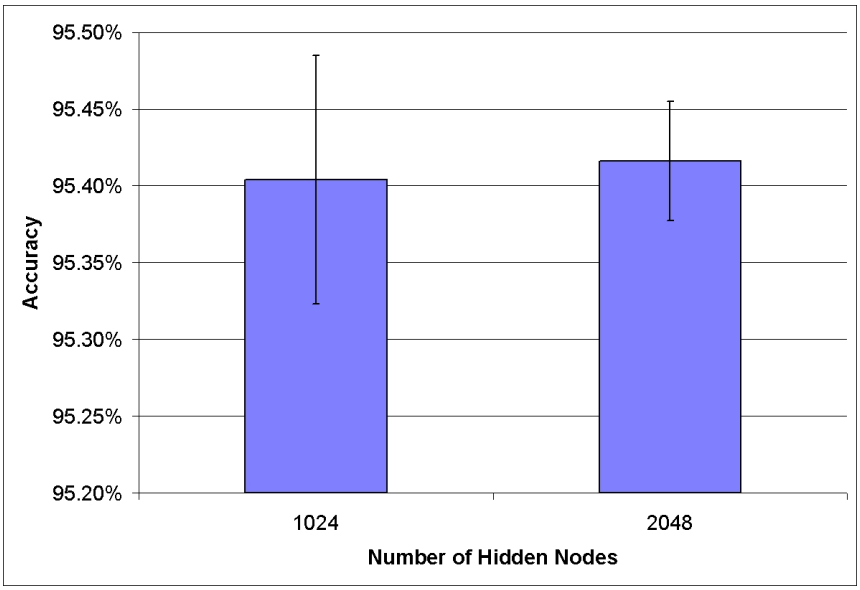

Figure 3.9: Mean accuracy with standard deviation using only 5% of the training set. The 2,048 hidden node ANN is chosen.

3.2.6 Performance Criteria

For every training set size in 3.2.2, and for every OTN size, five separate OTNs are trained using the training sets described in 3.2.4. For OCR there are a total of 20 experiments for each of the four OTN sizes except the 512 hidden node size (10 experiments) across four training sets and there are a total of 20-25 experiments per training set size (for a total of $20 \cdot 4 + 10$ or $20 \cdot 2 + 25 \cdot 2 = 90$ experiments). After every oracle learning epoch, character recognition accuracies are gathered using the hold-out set, and the respective OTN weights saved. The ANN most accurate on the hold-out set is then tested on the test set for a less biased measure of the OTN's performance. Finally, the five test set results from the five OTNs performing best on the hold-out set are averaged for the final performance metric.

Chapter 4

RESULTS AND ANALYSIS

4.1 Automated Speech Recognition Results

Figures 4.1-4.4 summarize the results of oracle learning for ASR by comparing each OTN with its standard-trained counterpart for each training set. The graphs show both error and error bars representing 2 standard deviations on both sides of the mean. They also give the decrease in error for each comparison. Decrease in error is calculated according to:

$$1 - \frac{Error_{otn}}{Error_{std}} \tag{4.1}$$

The 4,000 utterance case is shown in figure 4.1. This case is especially significant because it represents a situation in which the standard trained ANNs have sufficient data to solve the problem. Despite this, oracle learning yields better results because the OTNs have less error in each case. In the 50 hidden node case, the difference between standard and OTN accuracies is probably insignificant, but the standard deviation of the OTN is appreciably smaller. In the 100 hidden node case, the high end of the OTN's error bar is slightly above the standard trained ANN, and its standard deviation is larger. In a real world situation, choosing one or the other yields a tradeoff. The OTN has a lower error on average, but is more likely to produce ANNs

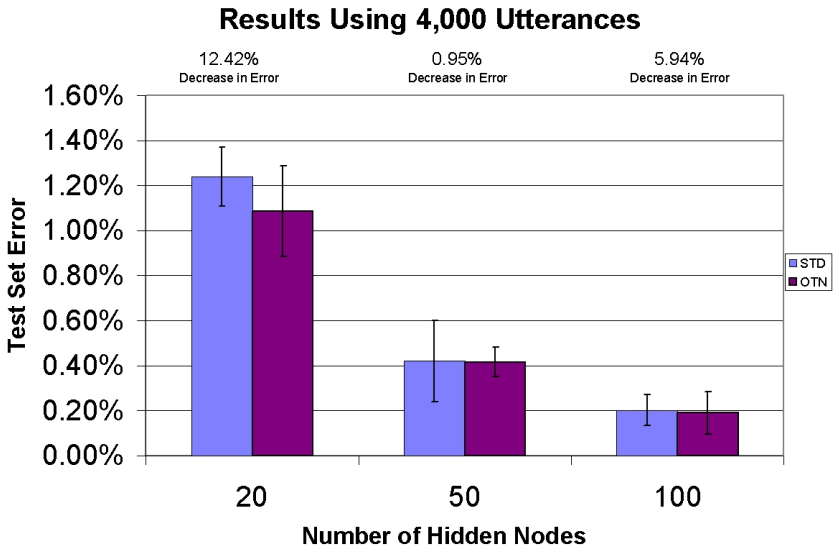

Figure 4.1: Mean OTN and standard test set accuracies using the 4,000 utterance oracle ANN. The OTN is the winner for the 20 hidden node case and still more accurate in both the 50 and 100 hidden node cases, but only slightly and with a greater standard deviation in the 100 hidden node case.

worse than the standard method. The standard trained ANN has a smaller standard deviation and better worst case behavior, but attains a consistently higher error. The 20 hidden node OTN shows the most significant improvement over its standard counterpart, yielding over a 12% decrease in error. Figure 4.2 describes oracle learning's behavior on ASR given only 1,000 utterances for training the oracle and the standard ANNs. The only exception to choosing oracle learning here is once again in the 100 hidden node case. The standard trained ANNs yield a lower error, but because of their greater standard deviation, they are more likely to produce ANNs worse than those derived through oracle learning. Still, oracle learning is again preferred overall in the 1,000 utterances case because it consistently produces acceptable and even superior results for the smaller ANNs. The 1000 utterance example is the only case where the 20 hidden node OTNs do not exhibit the greatest decrease in error, this

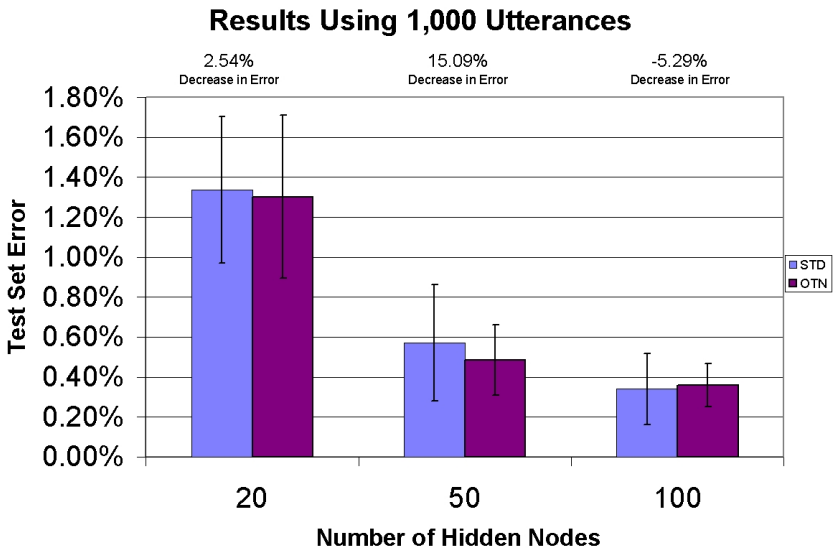

Figure 4.2: Mean OTN test set accuracies for OTNs trained using the 1,000 utterance oracle ANN compared to standard ANNs trained on the 1,000 utterance training set. The error bars show two standard deviations on both sides of the mean. In this case, the OTN is preferred in the 20 and 50 hidden node cases and although it has a higher error in the 100 hidden node case, its standard deviation is smaller.

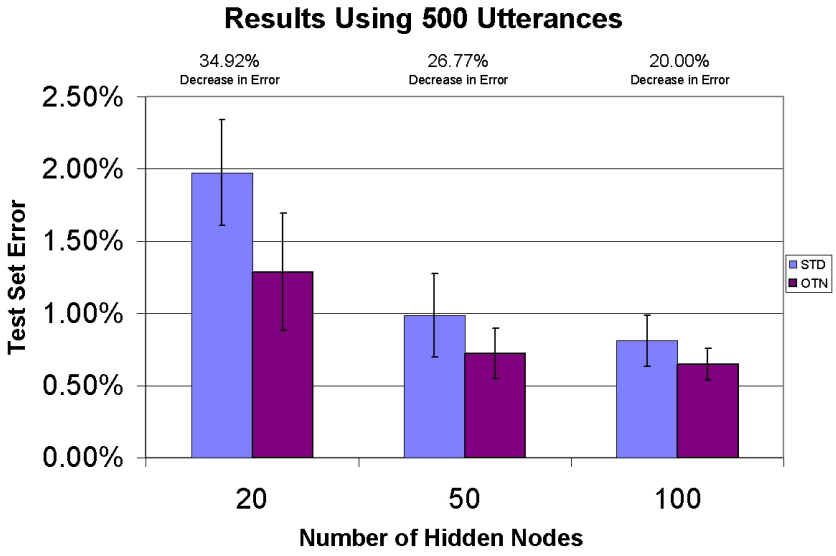

Figure 4.3: Mean OTN test set accuracies for OTNs trained using the 500 utterance oracle ANN compared to standard ANNs trained on the 500 utterance training set. The error bars show two standard deviations on both sides of the mean. In this case, the OTNs are the choice for every size ANN.

may be simply due to not having enough experiments (only five per ANN for that single comparison). The OTNs in the 500 utterances experiment (see figure 4.3) are the better ANNs in every case (20, 50, and 100 hidden nodes). Finally, the 150 utterances case (using 1% of the training data to train the oracle and the standard ANNs, see figure 4.4) again demonstrates oracle learning's ability to produce smaller ANNs that are more accurate than their standard trained counterparts.

In order to demonstrate how well the OTNs retain their oracles' accuracies, figures 4.5-4.7 show how similar each sized OTN is on average to its corresponding oracle's accuracy given increasingly more unlabeled versus labeled data. *Similarity* is simply the accuracy of the OTN or ANN divided by the accuracy of the oracle. As the amount of labeled training data available decreases, the general trend is for the OTNs to remain more similar to their oracles than the standard trained ANNs. For instance,

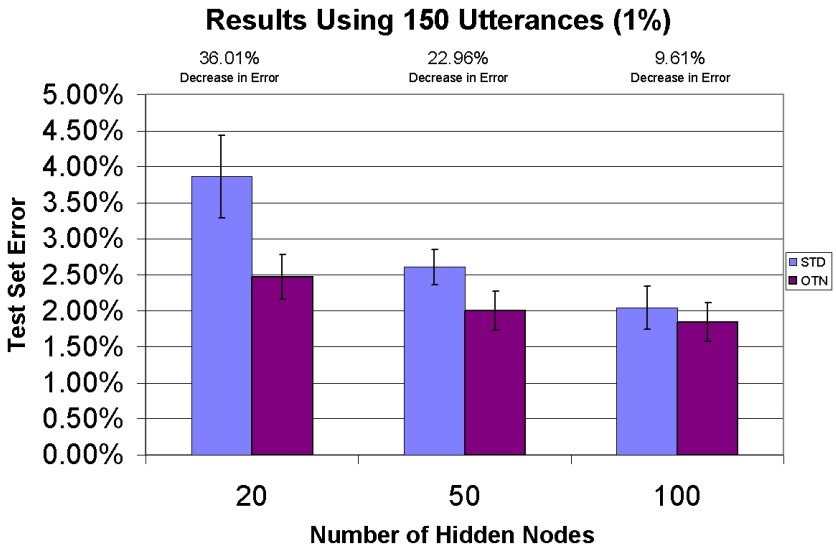

Figure 4.4: Mean OTN test set accuracies for OTNs trained using the 150 utterance oracle ANN compared to standard ANNs trained on the 150 utterance training set. The error bars show two standard deviations on both sides of the mean. The OTNs are again winners in this case.

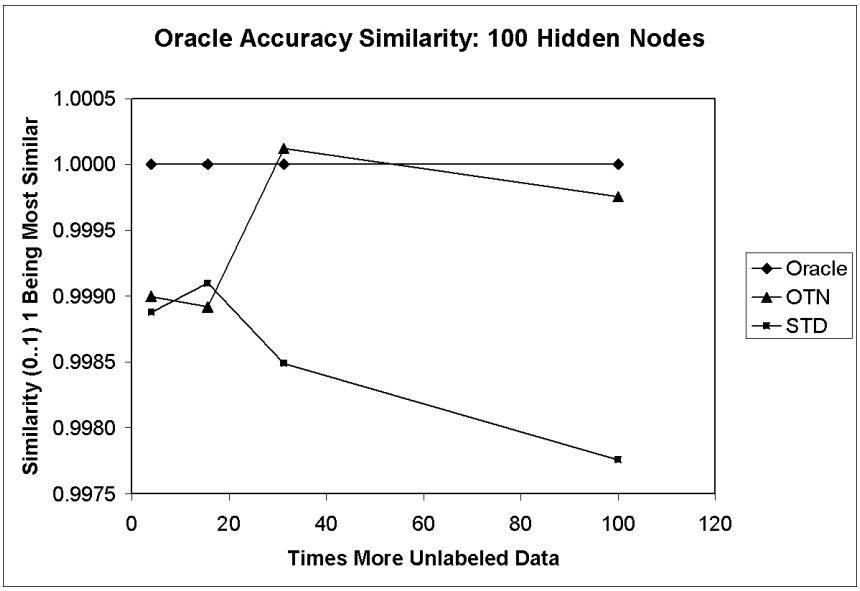

Figure 4.5: Oracle similarity for 100 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

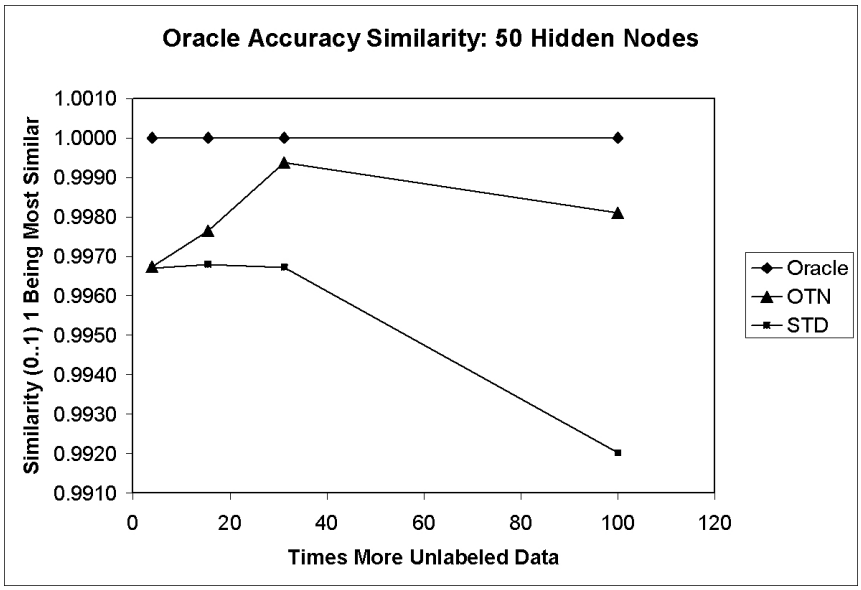

Figure 4.6: Oracle similarity for 50 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

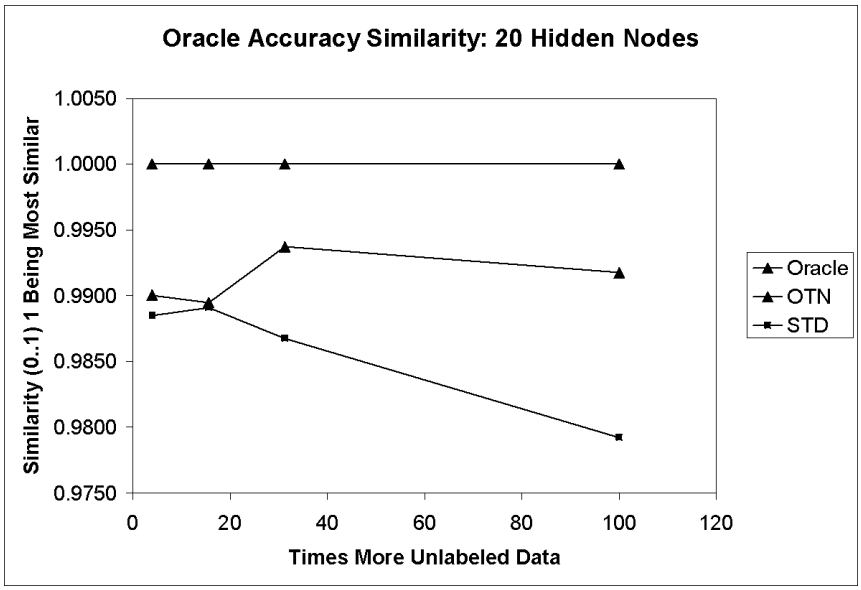

Figure 4.7: Oracle similarity for 20 hidden node OTN and standard trained ANN given increasing amounts of unlabeled versus labeled data.

in the 100 hidden node case (see figure 4.5), the OTNs and standard ANNs are similar until there are 30 times more unlabeled than labeled data. The other figures demonstrate similar behavior. The standard trained ANNs and OTNs begin being fairly equal, and then diverge as the amount of hand-labeled data becomes smaller compared to the amount of oracle-labeled data. The figures give evidence that as the amount of available labeled data decreases without a change in the amount of oracle-labeled data, oracle learning yields more and more improvement over standard training. This is probably due to the OTNs always having the same large amount of data to train on. They experience far more data points, and even though they are labeled by an oracle instead of by hand, the quality of the labeling is sufficient to exceed the accuracy attainable through training on only the hand labeled data.

Tables 4.1-4.4 present averages across training set sizes for a given OTN size and averages across OTN sizes for a given training set size. Tables 4.1 and 4.2 show decreases in error with respect to standard training. Table 4.1 gives the decrease in error using a given OTN size when averaged across the four training set sizes. The table suggests that oracle learning improves more over standard training as the size of the OTN decreases. Table 4.2 shows the decrease in error for a given training set size when averaged across the three OTN sizes. Here it appears that decreasing the amount of available hand-labeled data—thus increasing the relative amount of unlabeled data—yields greater improvements for oracle learning. The average decrease in error using oracle learning instead of standard methods is 15.16% averaged over the 60 experiments.

Tables 4.3 and 4.4 give average oracle similarites. Table 4.3 shows how oracle similarity varies for a given OTN size when averaged across training set sizes. Oracle similarity increases as the size of the OTN increases. Table 4.4 demostrates how the amount of hand-labeled data used to train the oracle and standard-trained ANNs

# Hidden Nodes	% Avg. Decrease in Error	Num. Experiments
20	21.47	20
50	16.44	20
100	7.56	20
Avg	**15.16**	

Table 4.1: Average decrease in error compared to standard methods for each of the three OTN sizes averaged across the four training set sizes.

# Utterances	% Avg. Decrease in Error	Num. Experiments
150	22.86	15
500	27.23	15
1000	4.11	15
4000	6.44	15
Avg	**15.16**	

Table 4.2: Average decrease in error compared to standard methods for each of the four training set sizes averaged across the three OTN sizes.

affects oracle similarity. As the amount of hand-labeled data decreases, the OTNs better approximate their oracles. Average oracle similarity across the 60 experiments is 0.9964.

4.2 Automatic Speech Recognition Analysis

The results above provide evidence that oracle learning can be beneficial when applied to ASR. With only one exception, the OTNs have less error than their standard trained counterparts. Oracle learning's performance improves with respect to standard training if either the amount of labeled data or OTN size decreases. Therefore,

# Hidden Nodes	Oracle Similarity	Num. Experiments
20	.9916	20
50	.9980	20
100	.9994	20
Avg	**.9964**	

Table 4.3: Oracle similarity for each of the three OTN sizes averaged across the four training set sizes.

# Utterances	Oracle Similarity	Num. Experiments
150	.9971	15
500	.9977	15
1000	.9953	15
4000	.9953	15
Avg	**.9964**	

Table 4.4: Average OTN oracle similarity for each of the four training set sizes averaged across the three OTN sizes.

for a given ASR application with only a small amount of labeled data, or given a case where an ANN of 50 hidden nodes or smaller is required, oracle learning is particularly appropriate. The 20 hidden node OTNs are an order of magnitidue smaller than their oracles and are able to maintain 99.16% of their oracles' accuracy averaged over the training set sizes with 21.47% less error than standard training. On average, oracle learning results in a 15.16% decrease in error compared to standard methods. Oracle learning also allows the smaller ANNs to retain 99.64% of their oracles' accuracy on average.

Why does oracle learning perform so well? The obvious answer is that there are enough oracle-labeled data points for the OTNs to effectively approximate their oracles. Since the larger, standard-trained oracles are always better than the smaller, standard-trained ANNs, OTNs that behave *like* their oracles are usually more accurate than their standard-trained equivalents.

Another reason for oracle learning's success is that the OTNs have a larger training set than the standard-trained ANNs. As stated above, even though the OTN training set is not hand-labeled, the oracle labels are accurate enough to produce favorable results. Apparently a large, oracle-labeled training set outperforms smaller, hand-labeled sets—especially as the hand-labeled set continues to decrease in size. This also explains why oracle learning's performance appears to increase over standard results in the experiment as the amount of hand-labeled data decreases.

There are two other trends in the results worth treating. The first deals with oracle similarity. The larger the ANN, the better it retains oracle similarity. The obvious reason for this is that larger ANNs overfit more to their training sets. Since the OTN training sets are oracle-labeled, the more an ANN overfits them, the more similar they are to their oracles. This is one of the gains of oracle learning—overfitting is actually preferable. The second trend is that as the size of the OTN decreases, its gains over standard-training increase. This may be because the 20 hidden node ANN has enough oracle-labeled data to reach its potential whereas the 100 hidden node ANN does not. Methods of testing how the amount of oracle-labeled data affects oracle learning results are discussed further in section 5.2.

4.3 Optical Character Recognition Results

Figures 4.8-4.11 summarize the results of oracle learning for OCR by comparing each OTN with its standard-trained counterpart. The graphs show both error and error bars representing two standard deviations on both sides of the mean. In every case, oracle learning produces OTNs that exhibit less error than the standard ANNs for OCR.

Figures 4.12-4.15 show how oracle similarity varies given less labeled training data. As the amount of available labeled training data decreases, the OTNs become more similar to their oracles whereas the standard trained ANNs diverge from them.

Finally, tables 4.5-4.8 present averages across training set sizes for a given OTN size and averages across OTN sizes for a given training set size. The averages given at the bottom of each table are weighted by the number of experiments in each entry since that number varies. Tables 4.5 and 4.6 show decreases in error with respect to standard training. Table 4.5 gives the decrease in error using a given OTN size when averaged across the four training set sizes. The table suggests that oracle learning improves more over standard training as the size of the OTN decreases. Table 4.8

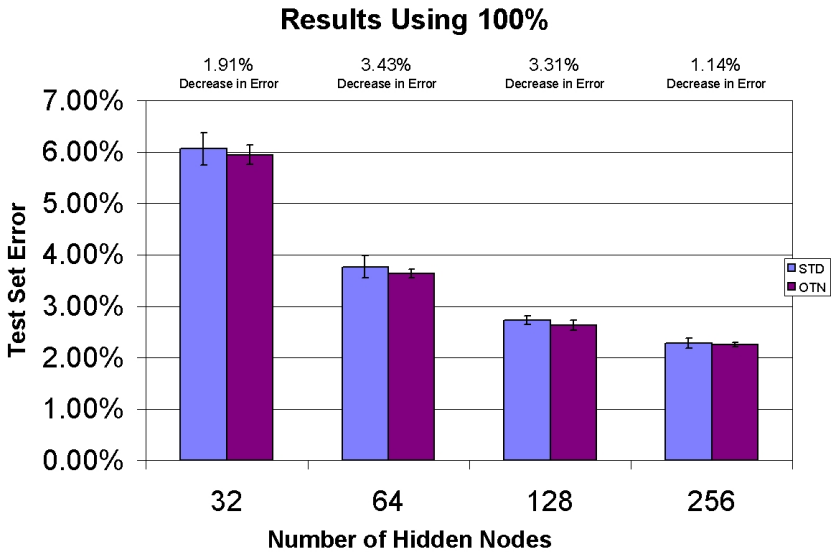

Figure 4.8: Mean OTN test set accuracies with standard deviation for OTNs trained using the entire OCR training set.

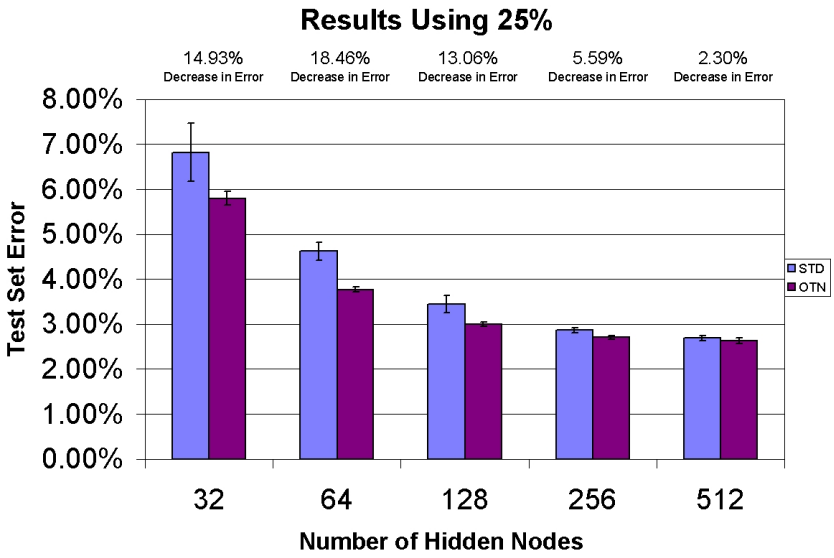

Figure 4.9: Mean OTN test set accuracies with standard deviation for OTNs trained using the 25% of the OCR training set.

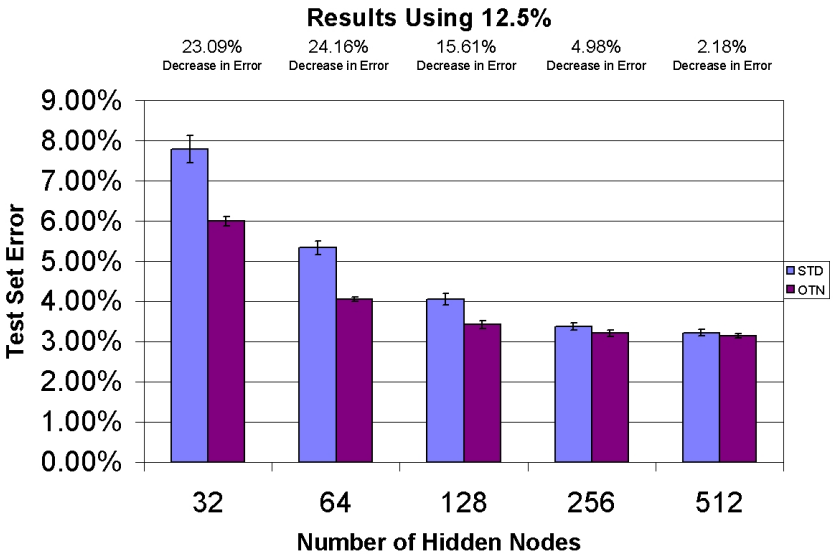

Figure 4.10: Mean OTN test set accuracies with standard deviation for OTNs trained using the 12.5% OCR training set.

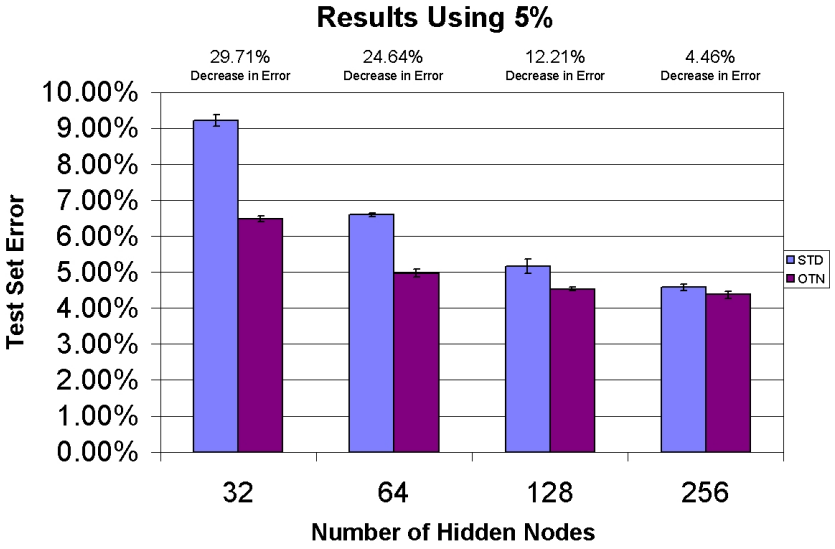

Figure 4.11: Mean OTN test set accuracies with standard deviation for OTNs trained using the 5% OCR training set.

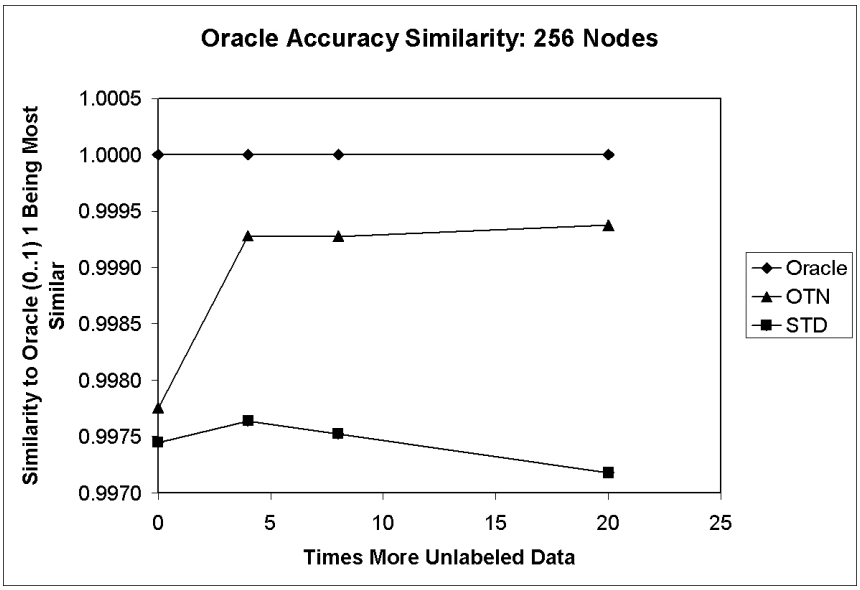

Figure 4.12: Oracle similarity for 256 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

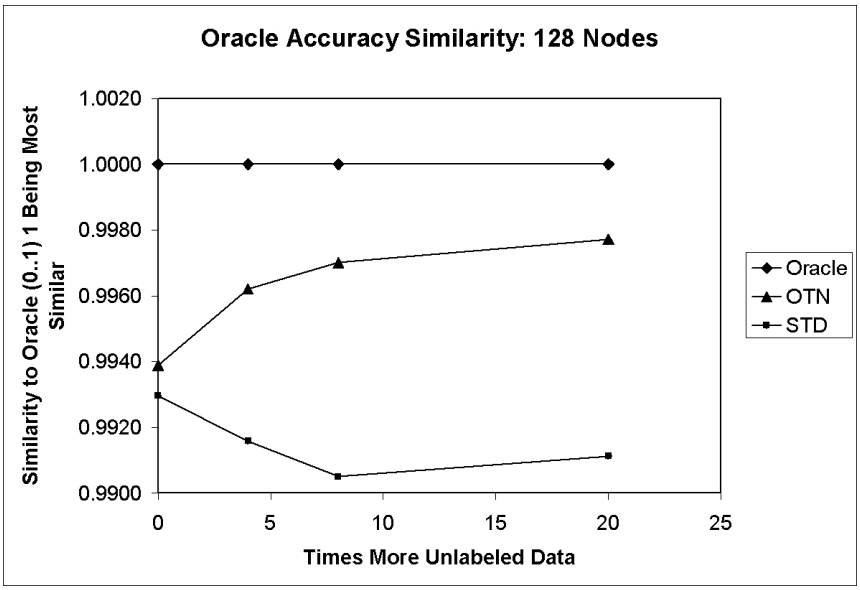

Figure 4.13: Oracle similarity for 128 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

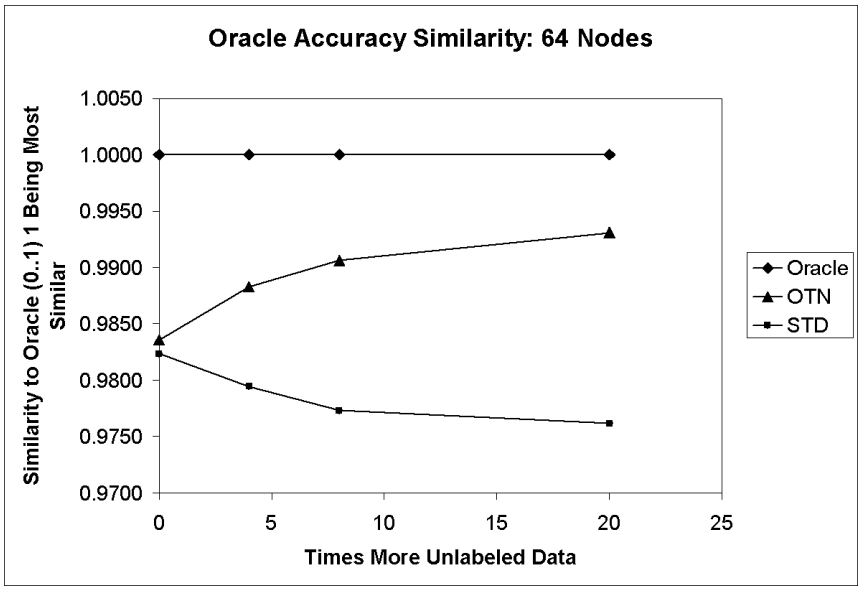

Figure 4.14: Oracle similarity for 64 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

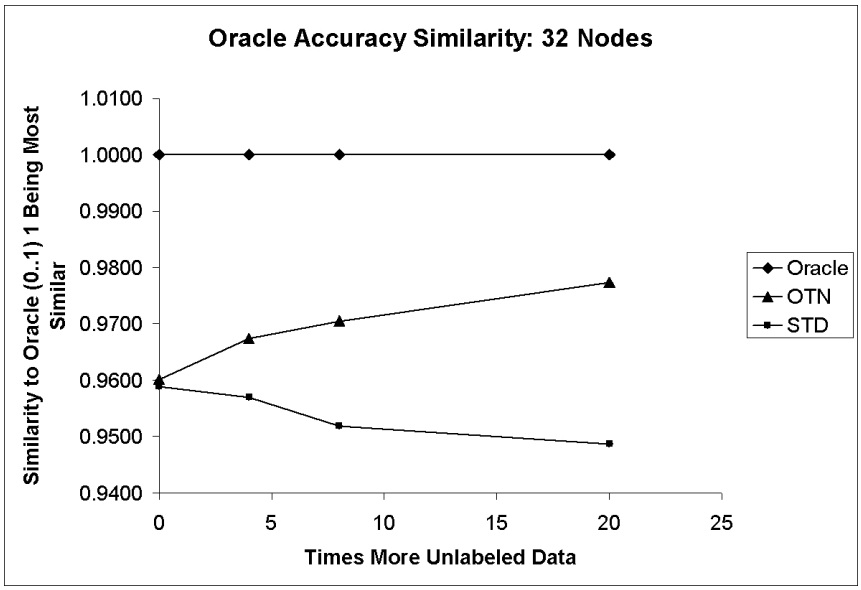

Figure 4.15: Oracle similarity for 32 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

# Hidden Nodes	% Avg. Decrease in Error	Num. Experiments
32	17.41	20
64	17.67	20
128	11.05	20
256	4.04	20
512	2.24	10
Avg (weighted)	**11.40**	

Table 4.5: Average decrease in error over standard methods for four of the OTN sizes averaged across the four training set sizes.

% of Training Set	% Avg. Decrease in Error	Num. Experiments
5	17.75	20
12.5	14.00	25
25	10.87	25
100	2.45	20
Avg (weighted)	**11.40**	

Table 4.6: Average decrease in error compared to standard methods for each of the four training set sizes averaged across the three OTN sizes.

shows the decrease in error for a given training set size when averaged across the three OTN sizes. Here it appears that decreasing the amount of available hand-labeled data—thus increasing the relative amount of unlabeled data—yields greater improvements for oracle learning. The average decrease in error using oracle learning instead of standard methods is 11.40% averaged over the 90 experiments.

Tables 4.7 and 4.8 give average oracle similarites. Table 4.7 shows how oracle similarity varies for a given OTN size when averaged across training set sizes. Oracle similarity increases as the size of the OTN increases. Table 4.8 demostrates how the amount of hand-labeled data used to train the oracle and standard-trained ANNs affects oracle similarity. As the amount of hand-labeled data decreases, the OTNs better approximate their oracles. Average oracle similarity across the 60 experiments is 0.9895.

# Hidden Nodes	Oracle Similarity	Num. Experiments
32	.9688	20
64	.9889	20
128	.9962	20
256	.9989	20
512	.9999	10
Avg (weighted)	**.9895**	

Table 4.7: Oracle similarity for four of the OTN sizes averaged across the four training set sizes.

% of Training Set	Oracle Similarity	Num. Experiments
5	.9919	20
12.5	.9914	25
25	.9902	25
100	.9838	20
Avg (weighted)	**.9895**	

Table 4.8: Oracle similarity for each of the four training set sizes averaged across the three OTN sizes.

4.4 Optical Character Recognition Analysis

In the OCR experiment the OTNs are preferable to their standard trained counterparts in every case. The 32 hidden node OTNs are two orders of magnitude smaller than their oracles and are able to maintain 96.88% of their oracles' accuracy while improving 17.41% in average error over standard training. Overall, oracle learning decreases standard training's average error by 11.40% while maintaining 98.95% of the oracles' accuracy. As with the ASR results, the smaller OTNs demonstrate greater improvement over standard training on the smaller OTNs than the larger, and are also more effective when less hand-labeled data is available. The OCR results imply that even in the absence of a decoder, a large, oracle-labeled training set can yield higher accuracies than smaller, hand-labeled sets.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The purpose of the given research is to present and defend oracle learning as a method of producing smaller ANNs that (1) retain similarity to the most accurate ANN solution to a given problem, and (2) are more accurate than their standard trained counterparts. The first two chapters present oracle learning and the third and fourth present the experimental evidence defending its ability to accomplish its goals. On automatic spoken digit recognition oracle learning decreases the average error by 15.16% over standard training methods while still maintaining, on average, 99.64% of the oracles' accuracy. For optical character recognition, oracle learning results in a 11.40% decrease in error over standard methods, maintaining 98.95% of the oracles' accuracy, on average. The results also suggest oracle learning works best under the following conditions:

1. The size of the OTNs is small.

2. The amount of available hand-labeled data is small.

5.2 Future Work

One important trend to consider is how oracle learning's performance varies if there are more or less available unlabeled data given a sufficient amount of hand-labeled data. Does oracle learning's accuracy decrease significantly if the training set used by the OTNs is smaller than the ones described in 3.1 and 3.2? Does oracle learning's decrease in error over standard methods continue to improve if the amount of oracle-labeled data is significantly greater than the sets used? One way to observe this trend is to remove data from the 15,322 utterance training set used in the above experiments. Consider the case where 500 utterances are used to train the oracle. Instead of using the 500 utterance oracle to label all 15,000 data points for training the OTNs, labeling only 4,000 demonstrates how well oracle learning does with less oracle data. If oracle learning scales, there will be a significant drop in accuracy. In fact, five such experiments have already been conducted and the averaged result suggests a 12% improvement in accuracy when using 15,000 instead of 4,000 utterances of oracle-labeled data. The opposite approach, adding more oracle-labeled data, also has merit—especially since it will show improvement over the results in Chapter 4 if oracle learning scales. If larger OTNs need more data in general than smaller OTNs to reach their potentials, increasing the amount of available data will allow the larger OTNs to obtain a greater relative improvement over standard-trained ANNs than is currently observed. Both approaches will be considered further for future research.

Another interesting question is whether or not successively smaller OTNs can be trained on each other. For example, a 100 hidden node OTN can be used as an oracle to train 50 and 20 hidden node OTNs with possibly improved performance over the 50 and 20 hidden node OTNs trained with a larger oracle. The reasoning behind this is that the function the 100 hidden node OTN learned is by definition simpler than

its larger oracle ANN, and therefore may be easier for an even smaller ANN to learn. Unfortunately, there is still a degradation in accuracy (except for a few cases) creating the 100 hidden node OTN, and therefore it is an inferior oracle. Preliminary results training 20 hidden node OTNs using 100 hidden node OTNs were also discouraging.

The outputs for the OTNs in the given research used a sigmoidal transfer function with cross-entropy as the objective function. Another future experiment will instead use a linear transfer function on the outputs with sum squared error as the objective function since this setup is more common for function approximation. It may be oracle learning does even better as a function approximation problem than as a classification problem.

One of the interesting results in section 4.3 that has not yet been discussed is that oracle learning results in a slight improvement over standard training even when using the same training set for both the standard- and oracle-trained networks. The only difference between the two training sets is that one is oracle-labeled, and the other hand-labeled. The improvement is small, only 2.44%, and therefore may be due only to statistical variation. However, the fact that it is consistent across all four OTN sizes begs the question of whether or not oracle learning yields better results for a reason other than simply the increase in available data. Since the only difference is the labels, it may be that oracle learning produces targets that are easier for backpropagation-trained ANNs to learn given a set amount of hidden nodes. One major difference between oracle and hand labels is that oracle labels always result in less back-propagated error because the targets are always either greater than 0, or less than 1. Hand labels for classification problems are always 0 or 1. Less error may result in a function that is easier for backpropagation to learn. Caruana presents arguments in [24, 25, 26] that for a given function $f(x)$, there may be another function $g(x)$ that still solves the given problem, but is easier to learn for backpropagation.

For ASR and OCR, it may be that learning to approximate the oracle is a function $g(x)$ that is easier to learn than the true, 0-1 classification function $f(x)$. One way to test this hypothesis is to conduct more experiments where both the OTNs and the standard-trained ANNs use the same training set. If oracle learning does indeed produce easier targets for backpropagation, perhaps it can be determined through further analysis why the function is easier. Knowing why the oracle's function is easier for backpropagation may lead to other methods of converting a standard 0-1 classification function to an easier function, resulting in oracle learning quality results without an oracle. The same information could also lead to novel learning algorithms that improve over standard training methods.

An interesting area where oracle learing may provide an appropriate solution is approximating a given function that normally uses multiple models, each effectively approximating only specific parts of the function's range. Usually several of these models would be used together for a complete solution. It may be possible, however, to use oracle learning to reduce the multiple-model solution to a single OTM. Each of the original models would be used as an oracle to label those parts of the training set that correspond to that model's "expertise." As an example, consider ASR given varying levels of noise. One standard solution is to train several ANNs, each on a different noise level, requiring preprocessing at runtime to determine the noise level of the speech sample to be recognized. The oracle learning solution would be to use the original ANNs to label data from their respective noise levels, constructing a large training set consisting of data from each noise level labeled by an ANN trained on that noise level. The resulting OTN may be able to achieve similar performance to the original ANNs without needing any preprocessing to determine the level of noise and with one instead of multiple ANNs.

Bibliography

[1] Mitchell, T.M. (1997). *Machine Learning.* Boston, MA: McGraw Hill.

[2] Quinlan, J.R. (1979). Discovering rules by induction to form large collections of examples. In D. Michie (Ed.), *Expert Systems in the Micro Electronic Age.* 168–201, Edinburgh, UK: Edinburgh University Press.

[3] Quinlan, J.R. (1983). Learning efficient classification procedures and their application to chess end games. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach.* 463–482, San Mateo, CA: Morgan Kaufmann.

[4] Rumelhart, D., Widro, B., & Lehr, M. (1994). The basic ideas in neural networks. In *Commnications of the ACM.* 37(3), 87–92.

[5] Lecun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., & Jackel, L.D. (1989). Backpropagation applied to handwritten zip code recognition. In *Neural Computation.* 1(4), 541–551.

[6] Le Cun Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., & Jackel, L.D. (1990). Handwritten digit recognition with a back-propagation network. In D.S. Touretzky, (Ed.), *Advances in Neural Information Processing Systems 2.* 396–404, San Mateo, CA: Morgan Kaufmann.

[7] Waibel, A. (1989). Consonant recognition by modular construction of large phonemic time-delay neural networks. In *Proceedings of the 6th IEEE International Conference on Acoustics, Speech, and Signal Processing.* 112–115, Glasgow, Scotland.

[8] Waibel, A., Hanazawa, T., Hinton, G., Shikako, K., & Lang, K. (1989). Phoneme recognition using time-delay neural networks. In *IEEE Transactions on Acoustics, Speech and Signal Processing.* 37(3), 328–339.

[9] Lang, K.J., Waibel, A.H., & Hinton, G.E. (1990). A time-delay neural network architecture for isolated word recognition. In *Neural Networks.* 3(1), 23–43.

[10] Cottrell, G.W. (1990). Extracting features from faces using compressing networks: face, identity, emotion and gender recognition using holons. In D.S. Touretzky (Ed.), *Connection Models: Proceedings of the 1990 Summer School.* 328–337, San Mateo, CA: Morgan Kaufmann.

[11] Duda, R.O., Hart, P., & Stork, D., (2001). *Pattern Classification.* New York: John Wiley & Sons.

[12] Lewis, D. (1991). *Representation and Learning in Information Retrieval.* (Ph.D. thesis), (COINS Technical Report 91–93). Dept. of Computer and Information Science, University of Massachusetts. Amherst, MA.

[13] Lang, K. (1995). Newsweeder: learning to filter netnews. In Prieditis, & Russell (Eds.), *Proceedings of the 12th International Conference on Machine Learning.* 331–339, Lake Tahoe, CA.

[14] Joachims, T. (1996). *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization.* (Computer Science Technical Report CMU-CS-96-118). Carnegie Melon University. Pittsburg, PA.

[15] Holland, J.H. (1962). Outline for a logical theory of adaptive systems. In *Journal of the ACM.* 9(3), 297–314.

[16] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Ann Arbor, MI: University of Michigan Press.

[17] Le Cun Y., Denker, J.S., & Solla, S.A. (1990). Optimal brain damage. In D.S. Touretzky, (Ed.), *Advances in Neural Information Processing Systems 2.* 598–605, San Mateo, CA: Morgan Kaufmann.

[18] Domingos P. (1997). Knowledge acquisition from examples via multiple models. In *Proceedings of the Fourteenth International Conference on Machine Learning.* 211–218, Nashville, TN.

[19] Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

[20] Zeng, X, & Martinez, T.R. (2000). Using a neural network to approximate an ensemble of classifiers. In *Neural Processing Letters.* 12(3), 225–237.

[21] Breiman, L. (1996). Bagging predictors. In *Machine Learning.* 24(2), 123–140.

[22] Craven, M.W., & Shavlik, J. W. (1993). Learning symbolic rules using artificial neural networks. In *Proceedings of the 10th International Conference on Machine Learning.* 73–80, Amherst, MA.

[23] Craven, M.W., & Shavlik, J.W. (1996). Extracting tree-structured representation from trained networks. In D.S. Touretzky, M.C. Mozer, & M. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8.* 24–30, Cambridge, MA: MIT Press.

[24] Caruana, R., Baluja, S., & Mitchell, T. (1996). Using the future to 'sort out' the present: rankprop and multitask learning for medical risk evaluation. In *Advances in Neural Information Processing Systems 8.* 959–965, Cambridge: MA: MIT Press.

[25] Caruana, R. (1996). Algorithms and applications for multitask learning. In *Proceedings of the Thirteenth International Conference on Machine Learning.* 87–95, Bari, Italy.

[26] Caruana, R. (1997). *Multitask Learning.* Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University. Pittsburg, PA.

[27] Leonard, G. R., & Doddington, G. (1993). TIDIGITS speech corpus, http://morph.lds.upenn.edu/Catalog/LDC93S10.html. Texas Instruments, Inc.