SCIMA 2003 - International Workshop on
Soft Computing Techniques in Instrumentation, Measurement and Related
Applications Provo, Utah, USA, 17 May 2003

Simplifying OCR Neural Networks with Oracle Learning

Joshua Menke and Tony Martinez
Department of Computer Science
Brigham Young University, Provo, UT, 84604
Email: josh@axon.cs.byu.edu, martinez@cs.byu.edu

Abstract* – Often the best model to solve a real world problem is relatively complex. The following 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 paper looks at oracle learning as applied to multi-layer perceptrons trained using standard backpropagation. 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.***

I. INTRODUCTION

As Le Cun, Denker, and Solla observed in [3], often the best *artificial neural network* (ANN) to solve a real-world problem is relatively complex. They point to the large ANNs Waibel used for phoneme recognition in [2] and the ANNs of Le Cun et al. with handwritten character recognition in [1]. "As applications become more complex, the networks will presumably become even larger and more structured" [3]. The following research presents the *oracle learning* algorithm, a training method that seeks 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 research is to increase the accuracy of these smaller, less resource intensive ANNs using oracle learning.

As an example consider designing an ANN for optical character recognition in a small, handheld scanner. The network has to be small, fast, and accurate. Now suppose the most accurate digit recognizing ANN given the available training data has 2048 hidden nodes, but the resources on the scanner allow for only 64 hidden nodes. One solution is to train a 64 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 64 hidden node ANN that does not suffer from nearly as significant a decrease in accuracy. Oracle learning uses the original 2048 hidden node ANN as an oracle to create as much training data as necessary using unlabeled character data. The oracle labeled data is then used to train a 64 hidden node network to approximate the 2048 hidden node network. The results in section IV show the oracle learning ANN retains 98.9% of the 2048 hidden node ANN's accuracy on average, while being $\frac{1}{32}$ the size. The resulting *oracle-trained network* (OTN) is almost 18% more accurate on average than the standard trained 64 hidden node ANN.

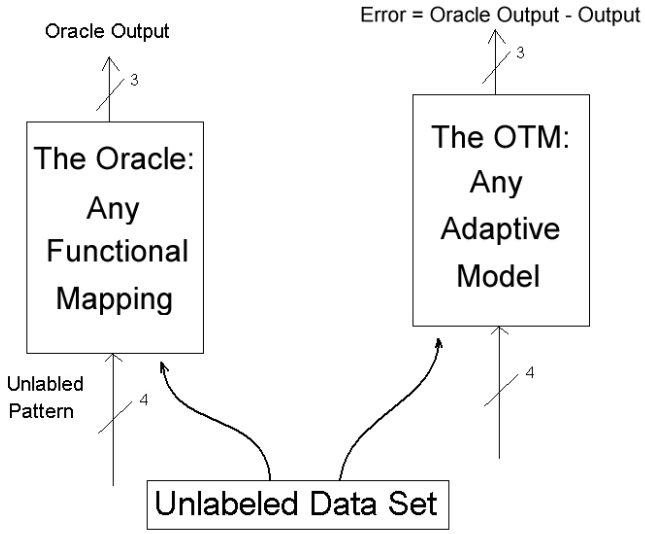

Fig. 1. Oracle Learning Summary

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 can be any machine learning model (e.g. an ANN, a nearest neighbor model, a bayesian learner, etc.). In fact, the oracle model can be any ar-

bitrary 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, 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 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:

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

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

$$\mathrm{OTN}(2048 \rightarrow 64)$$

reads "an OTN approximating an 2048 hidden node ANN with a 64 hidden node ANN."

The idea of approximating a more complex model is not new. A previous paper tested oracle learning's potential on speech recognition [4]. Domingos used Quinlan's C4.5 decision tree approach from [6] in [5] to approximate a bagging ensemble (bagging is a method of combining models, see [8] for details) and Zeng and Martinez used an ANN in [7] to approximate a similar ensemble (both using the bagging algorithm Breimen proposed in [8]). Craven and Shavlik used a similar approximating method to extract rules [9] and trees [10] 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 ..." [7]. While previous research has focused on either extracting information from neural networks [9],[10] or using statistically generated data for training [5], [7], the novel approach presented here and in the previous paper [4] is that currently unused, unlabeled data be labeled using the more complex model as an oracle.

II. ORACLE LEARNING

Oracle learning consists of the following 3 steps:

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

A. 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, a 2,048 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.

B. 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 [7], 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 *optical character recognition* (OCR), 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, automatic speech 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 [7].

C. 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.

$$\langle 0.27, 0.34, 0.45, 0.89, 0.29 \rangle \tag{1}$$

Now suppose the OTN outputs the following vector:

$$\langle 0.19, 0.43, 0.3, 0.77, 0.04 \rangle \tag{2}$$

The oracle-trained error is the difference between the target vector in 1 and the output in 2:

$$\langle 0.08, -0.09, 0.15, 0.12, 0.25 \rangle \tag{3}$$

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.

III. OPTICAL CHARACTER RECOGNITION EXPERIMENT

The following experiment serves to validate the effectiveness of oracle learning on a real-world problem. One 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.

A. 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. 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). 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.

B. Obtaining the Oracles

The OCR ANNs are feed-forward single hidden layer networks trained using standard backpropagation. For testing, the highest ANN output classifies the corresponding character. The ANN are trained and after each iteration, the weight configurations are saved for future testing. To determine the best size ANN (oracle ANN) for each of the four training sets, 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 2-5 graph the mean accuracy and give error bars representing two standard deviations in accuracy for the 5 ANNs averaged. In figure 2, 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) 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

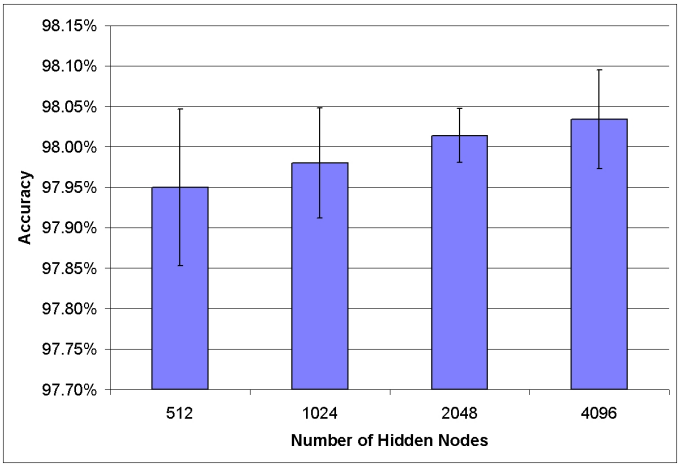

Fig. 2. Mean accuracy with standard deviation for ANNs using 100% of the training data. The 4096 hidden node ANN has the highest accuracy, but the 2048 hidden node ANN is nearly as accurate and varies less in its accuracy.

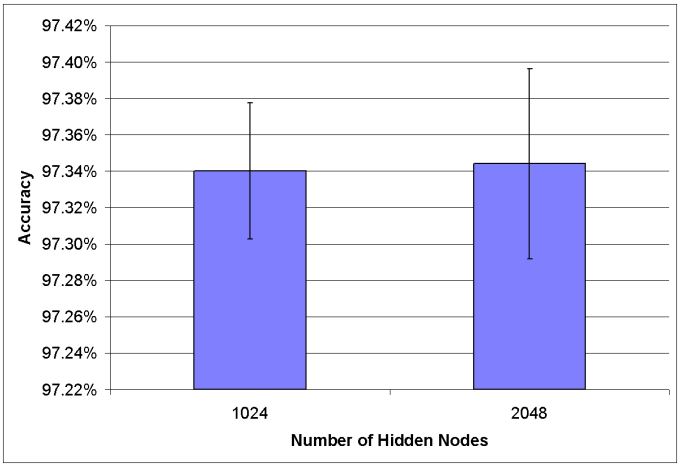

Fig. 3. Mean accuracy with standard deviation using 25% of the training data. The 1,024 hidden node ANN is the chosen oracle since its accuracy is almost identical to the 2,048 hidden node ANN, but it deviates slightly less from its mean.

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 5), 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.

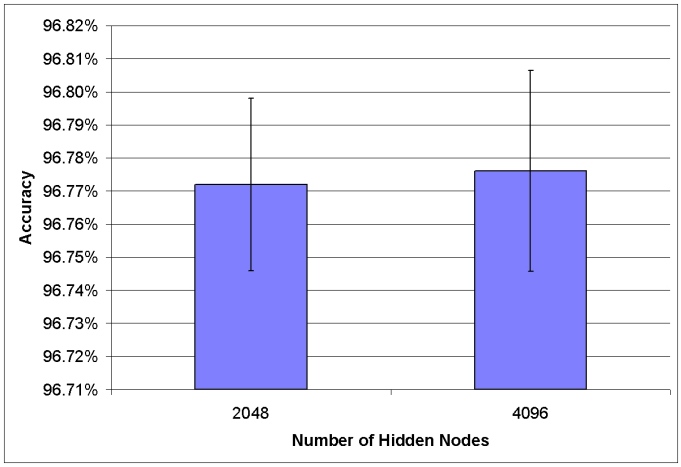

Fig. 4. Mean accuracy with standard deviation using 12.5% of the training data. Although almost identical, the 4,096 ANN is chosen for its slightly better performance.

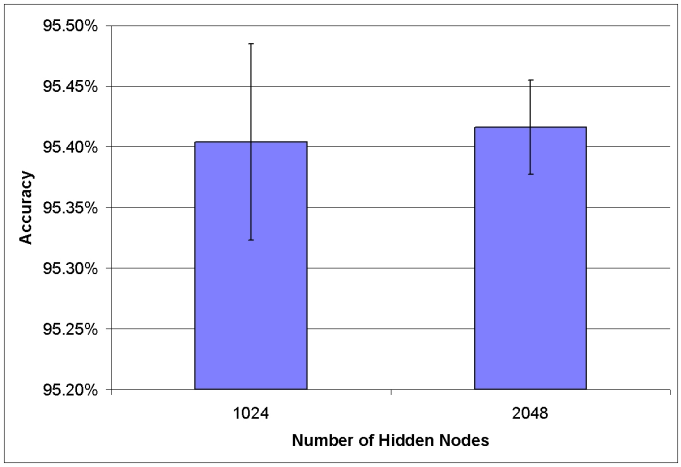

Fig. 5. Mean accuracy with standard deviation using only 5% of the training set. The 2,048 hidden node ANN is chosen.

C. Labeling the Data

For the next step a large training set is created by labeling the entire 400,000 character training set with each of the four oracles chosen in B. 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 section II.

D. Training the OTNs

The large OTN training sets described in C 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.

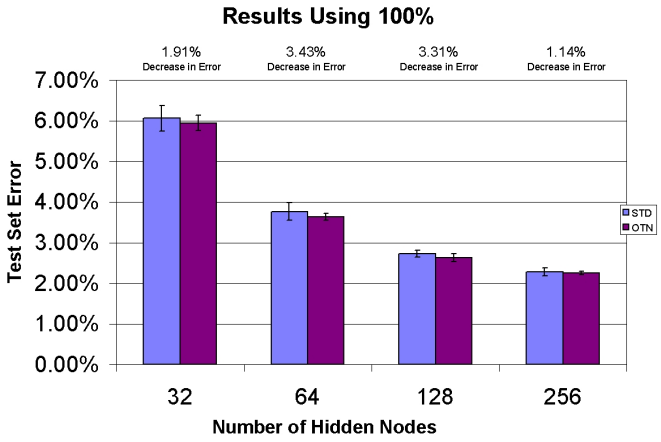

Fig. 6. Mean OTN test set accuracies with standard deviation for OTNs trained using the entire OCR training set.

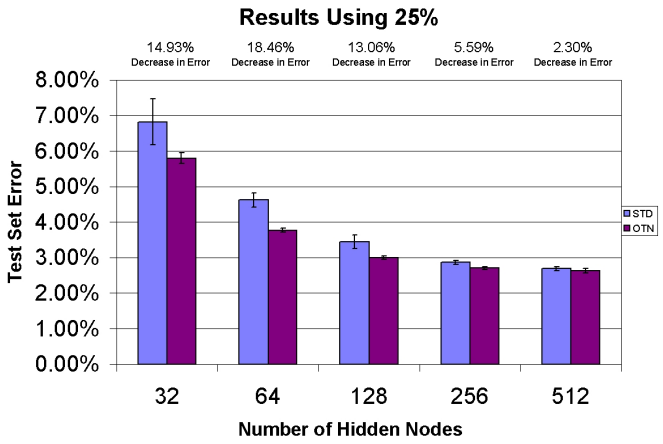

Fig. 7. Mean OTN test set accuracies with standard deviation for OTNs trained using the 25% of the OCR training set.

E. Performance Criteria

For every training set size in A, and for every OTN size, five separate OTNs are trained using the training sets described in C. 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.

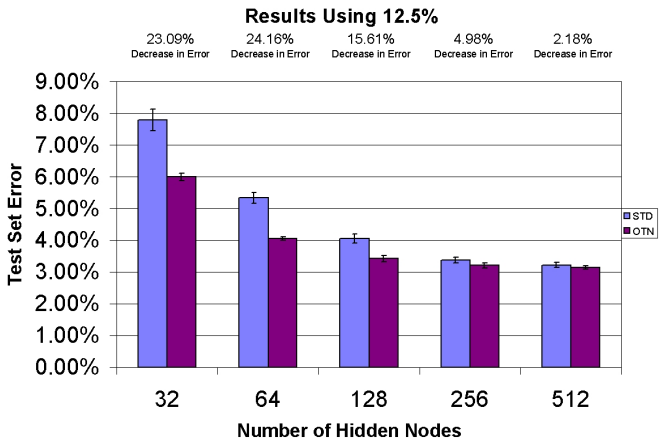

Fig. 8. Mean OTN test set accuracies with standard deviation for OTNs trained using the 12.5% OCR training set.

IV. RESULTS AND ANALYSIS

Figures 6-9 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 10-13 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 I-IV 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 I and II show decreases in error with respect to standard training. Table I 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 IV 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 III and IV give average oracle similarites. Table III 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 IV demostrates how

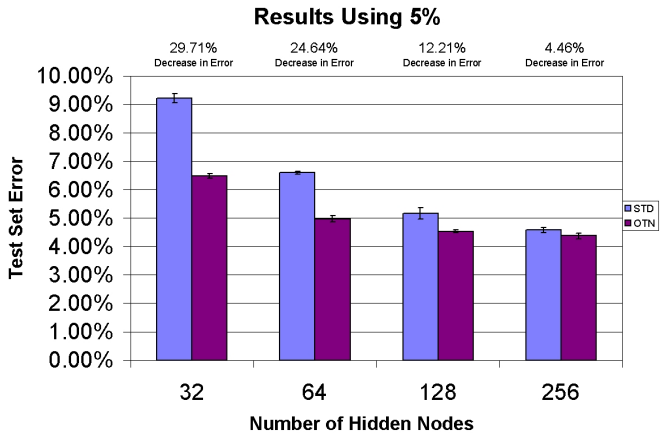

Fig. 9. Mean OTN test set accuracies with standard deviation for OTNs trained using the 5% OCR training set.

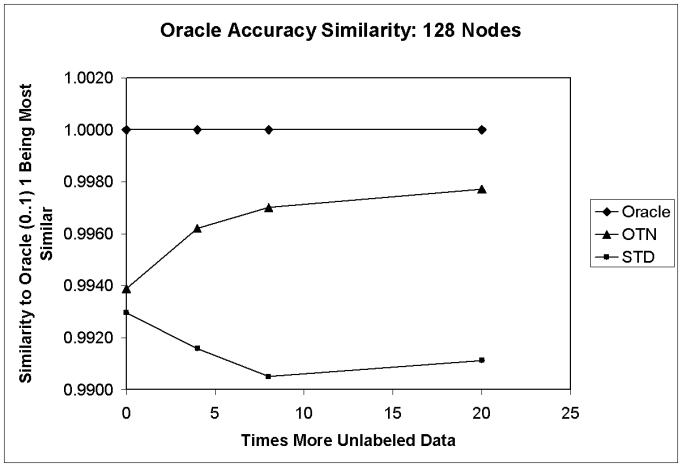

Fig. 11. Oracle similarity for 128 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

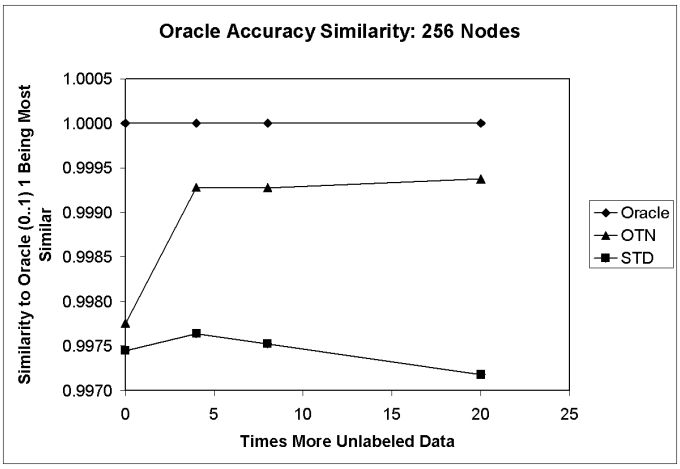

Fig. 10. Oracle similarity for 256 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

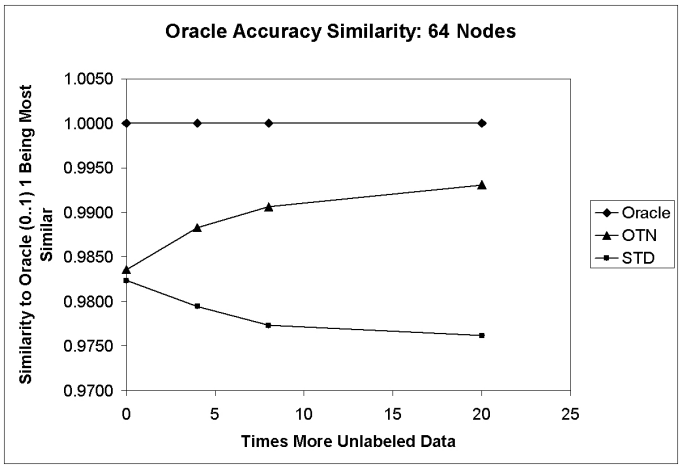

Fig. 12. Oracle similarity for 64 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

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.

The results above provide evidence that oracle learning can be beneficial when applied to OCR. As shown in the above results, the OTNs are preferable to their standard trained counterparts in every case. Oracle learning's performance improves with respect to standard training if either the amount of labeled data or OTN size decreases. Therefore, for a given OCR application with only a small amount of labeled data, or given a case where an ANN of 64 hidden nodes or smaller is required, oracle learning is particularly appropriate. 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. The smaller OTNs demonstrate greater improvement over standard training than the larger OTNs, and are also more effective when less hand-labeled data is available. The OCR results imply that a large, oracle-labeled training set can yield higher accuracies than smaller, hand-labeled sets.

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.

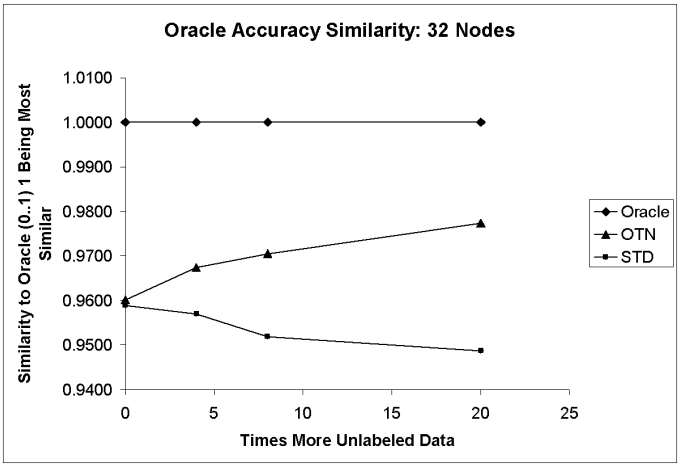

Fig. 13. Oracle similarity for 32 hidden node OTNs and standard trained ANNs given increasing amounts of unlabeled versus labeled data.

# 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 I

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 II

AVERAGE DECREASE IN ERROR COMPARED TO STANDARD METHODS FOR EACH OF THE FOUR TRAINING SET SIZES AVERAGED ACROSS THE THREE OTN SIZES.

# 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 III

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 IV

ORACLE SIMILARITY FOR EACH OF THE FOUR TRAINING SET SIZES AVERAGED ACROSS THE THREE OTN SIZES.

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 32 hidden node ANN has enough oracle-labeled data to reach its potential whereas the 256 hidden node ANN does not. Methods of testing how the amount of oracle-labeled data affects oracle learning results are discussed further in section VI.

V. CONCLUSION AND FUTURE WORK

A. 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. 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.

VI. 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 III? 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 add more oracle-labeled data since it will show improvement over the results in section IV 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 also allow the larger OTNs to obtain a greater relative improvement over standard-trained ANNs than is currently observed. This approach will be considered further for future research.

References

[1] 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.

[2] 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.

[3] 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.

[4] Menke, J., Peterson, A., Rimer, M., & Martinez, T. (2002). Network Simplification Through Oracle Learning. In *Proceedings of the IEEE Joint Conference on Neural Networks IJCNN'02*. 2482–2476, Honolulu, HA.

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

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

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

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

[9] 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.

[10] 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.