A New Pruning Algorithm for Neural Network Dimension Analysis

Devin Sabo, Xiao-Hua Yu

Abstract— **The choice of network dimension is a fundamental issue in neural network applications. An optimal neural network topology not only reduces the computational complexity, but also improves its generalization capacity. In this research, a new pruning algorithm based on cross validation and sensitivity analysis is developed and compared with three existing pruning algorithms on various pattern classification problems. Computer simulation results show the network size can be significantly reduced using this new algorithm while the neural network still maintains satisfactory generalization accuracy.**

I. Introduction

Artificial neural network has been successfully applied to solve many different problems, including dynamic system identification, pattern classification, and adaptive control. As we know, before a neural network can be employed, its dimension (i.e., number of layers, number of neurons in each layer, and how they are connected) must be predetermined. However, choosing the appropriate size of a neural network is a very difficult task and often comes down to a guess work. In general, if the network is too small, it may not be able to learn the desired relationship; on the other hand, if the network is too large, it yields more nodes, more weights, and more layers than necessary; and thus results in unnecessary arithmetic calculations and high computation cost. For real time applications, the reduction of network size may save us precious hardware implementation time. Also, the ability to generalize, or to produce accurate values for the inputs not included in the training dataset, is one of the major benefits of using neural networks. An oversized network may over-fit the training data, and has poor generalization ability for the testing data. Therefore, the choice of an optimal network dimension is an important issue in neural network design and applications.

There are two fundamental approaches to find the appropriate size of a neural network. The first one is to start with a small network and slowly add more connections to it until an appropriate stopping criterion is satisfied [11]. The network is first trained at its minimum size; then more weights/nodes can be added and the new network will be retrained. This process can be repeated until certain performance index is met. The difficulties of this approach include when to start the growing process, and where to add the new connections/nodes in the network. In addition, the above procedures may be very tedious and time-consuming.

The second approach is to start with a network that is knowingly too large for the problem, and then trim it down to the appropriate size. This is called "neural network pruning" and has been studied by many researchers in recent years ([1] [6] [9] [10] [12] [13]).

In this paper, a new pruning algorithm is investigated and compared with three existing ones, including the local sensitivity analysis method [13], the local variance sensitivity analysis [1], and the cross validation pruning algorithm [9]. This new algorithm combines the advantages of the above three methods, re-evaluates the network performance during pruning process, and iteratively prunes the neural network on a reduced set of connections if a pruning error occurs. A variety of data sets and various network configurations are studied in computer simulations. The results show that this new algorithm can significantly reduces the neural network size while still maintaining satisfactory generalization accuracy of the network.

II. The Neural Network Pruning Algorithm Studies

A typical neural network contains an input layer, an output layer, and one or more hidden layers. The number of outputs and inputs are usually fixed; while the number of hidden layers and number of hidden neurons in each hidden layer can be varied. In this research, we focus on the studies of pruning algorithms for multi-layer feedforward neural networks.

The simplest way to find the optimum network size is to use a brute force approach that produces all the combinations of networks within a desirable range, trains them, and then chooses the best one. This process is usually not an efficient way to solve the problem. Another approach, the weight decay method (or the penalty method), is based on the assumption that smaller weights in a network have relatively small effect on the output of a node, especially when surrounded by significantly larger weights. This method adds a penalty term to the objective function to be minimized so that these smaller weights can eventually be forced to zero. However, this approach may eliminate weights that are actually crucial to the overall architecture of the network and thus produce a network with poor performance. Also, the added penalty term may create additional local minima on the error surface during training.

Huynh and Setiono [9] introduced the concept of cross-validation. The whole dataset is divided into two parts, i.e., the training set T and the cross validation set C. The pruning

Manuscript received December 12, 2007. This work was supported in part by the Department of the Navy, Office of Naval Research, under Award # N00014-06-1-1111.

Devin Sabo was with the Department of Electrical Engineering, California Polytechnic State University, San Luis Obispo, CA 93407, USA. He is now with Lockheed Martin Corporation, CA, USA.

Xiao-Hua Yu is with the Department of Electrical Engineering, California Polytechnic State University, San Luis Obispo, CA 93407, USA (e-mail: xhyu@calpoly.edu).

978-1-4244-1821-3/08/$25.00©2008 IEEE

criterion is still based on the magnitude of each weight; however, a validation step is added to test the pruned network. At every pruning step, the performance of the network with reduced size is compared with the performance of the network before current pruning phase. Let the performance criterion (objective function) on set *T* and set *C* be J_{TR} and J_{CV}, respectively (where $J(\bullet)$ can be the root-mean-square error (RMS) for function approximation or identification problem, or the percentage of misclassified patterns for classification problem). After pruning, a smaller neural network is obtained and the error on set *T* and set *C* be J'_{TR} and J'_{CV}, respectively. If

$$(J'_{TR} + J'_{CV}) < (J_{TR} + J_{CV}) \tag{1}$$

i.e., the pruned network outperforms the un-pruned one; then the pruned network is accepted and the pruning process can be continued. Otherwise, the network is restored to the size before the current pruning step. Obviously, the use of an additional cross validation set at each phase of the pruning takes into account that pruning is meant to not only reduce the size of a network, but also improve the network generalization capacity.

Rather than focusing on the magnitude of the weights in the network, the sensitivity based approach attempts to find the contribution of each weight in the network and then prunes the weights that have the least effect on the objective function. Mozer and Smolensky [12] suggested that the sensitivity of each weight can be found by measuring the difference on the performance of the network with/without that weight, i.e.,:

$$\begin{aligned} s_{j,i} &= J(w_{j,i} = 0) - J(w_{j,i} = w^f_{j,i}) \\ &= J(\text{without } w_{j,i}) - J(\text{with } w_{j,i}) \end{aligned} \tag{2}$$

where $s_{j,i}$ is the sensitivity (with respect to the removal of connection $w_{j,i}$); $w_{j,i}$ is the weight of the neural network from node *i* to node *j*; $w^f_{j,i}$ is the final value of weight $w_{j,i}$ when training is finished; and $J(\bullet)$ is the objective function.

Calculating Eq. (2) directly may be very time-consuming. Karnin [10] found an effective way to approximate it for the back-propagation algorithm:

$$s_{j,i} = \sum_{t=0}^{T-1} [\Delta w_{j,i}(t)]^2 \ \frac{w^f_{j,i}}{\eta (w^f_{j,i} - w^i_{j,i})} \tag{3}$$

where *T* is the total number of iterations (training epochs) needed to minimize the objective function *J*; η is the learning rate and $\Delta w_{j,i}$ is the change on weight $w_{j,i}$ in one training iteration. The absolute value of the estimated sensitivity for each weight, $|s_{j,i}|$, is then compared with a pre-determined threshold to determine whether the weight should be pruned or not. Note that this algorithm relies heavily on the selection of a threshold which must be determined beforehand which may differ between data sets and applications. If the threshold is too high, too many weights will be pruned and the pruned network may not function as desired; but if the threshold is too small, no weights will be pruned at all. Also, in this method, all the sensitivities in the network are compared with the same threshold, i.e., they are treated equally for pruning.

Ponnapelli et al. [13] suggested that the sensitivities of weights should only be compared with those related with the same node in the same layer. Thus, the concept of local relative sensitivity index (LRSI) is defined as the ratio of the sensitivity of a particular weight and the sum of all the sensitivities of the weights that are connected to the same node from the previous layer:

$$LRSI_{j,i} = \frac{|s_{j,i}|}{\sum_{m=1}^{M} |s_{j,m}|} \tag{4}$$

where *M* is the total number of connections to node *j* from the previous layer. For each node, any weight that has a local sensitivity less than a threshold will be pruned:

$$LRSI_{j,i} \le \beta \tag{5}$$

Even though the choice of the threshold (i.e., β) still depends on the rule of thumb, it is now a percentage which is relatively easier to be chosen. Note that this algorithm only considers weight removal; node pruning is not included. Theoretically, if all the weights that are connected to a single node are pruned, then this node can also be eliminated. However, this may take several rounds of pruning and training so it may not be a feasible solution in practice.

Engelbrecht [6] proposed a modified approach to sensitivity analysis. Instead of using the values of sensitivities directly, Engelbrecht found the average sensitivity of a network parameter (e.g., weight or node) over all the patterns, and then developed a new measure called variance nullity. That is, if the variance of sensitivity of a network parameter over all the patterns (denoted by $\sigma^2_{\theta_k}$ for parameter θ_k) is close to zero and the average sensitivity (also over all the patterns) is small, then we conclude that this parameter has little or no effect on the output of the neural network over all patterns and therefore can be eliminated. The variance of sensitivity is defined as:

$$\sigma^2_{\theta_k} = \frac{\sum_{p=1}^{P} (s^{(p)}_{\theta_k} - \tilde{s}_{\theta_k})^2}{P-1} \tag{6}$$

where *P* is the total number of patterns under consideration and $\tilde{s}_{\theta_k}$ is the average sensitivity over all the patterns:

$$\tilde{s}_{\theta_k} = \frac{\sum_{p=1}^{P} s^{(p)}_{\theta_k}}{P} \tag{7}$$

The parameter variance nullity (PVN) for each parameter is then defined as:

$$\gamma_{\theta_k} = \frac{(P-1)\,\sigma^2_{\theta_k}}{\sigma^2_0} \tag{8}$$

where σ_0^2 is a small constant value related with hypothesis test $H: \sigma_{\theta_k}^2 < \sigma_0^2$ [6].

This algorithm allows for pruning of both nodes and weights, with each parameter having a separate formula for the sensitivity calculation. The extension of a sensitivity measurement to nodes (not just weights) allows for the possibility of finding a smaller network, and also decreases the number of times to retrain the network before obtaining its final size. Note that in this algorithm, the pruning decision is based on the hypothesis test *H*; thus choosing the appropriate value of σ_0^2 is crucial to the success of this algorithm.

As we discussed earlier, relying on one single value of σ_0^2 for the entire network can lead to problems. Fnaiech et. al. [1] suggested that parameters within the same layer should be considered "locally" rather than "globally", and defined a new pruning index called the local parameter variance nullity (LPVN). The PVN for all parameters in the same layer are summed up; then the LPVN for each parameter (which represents the relative importance of PVN of a parameter in the layer) can be obtained and used for pruning:

$$L\gamma_{\theta_k^{[l]}} = \frac{\gamma_{\theta_k^{[l]}}}{\sum_{k=1}^{K} \gamma_{\theta_k^{[l]}}} \tag{9}$$

where $L\gamma_{\theta_k^{[l]}}$ is the LPVN for layer *l*, and *K* is the total number of parameters in layer *l*.

As a summary, all the above algorithms have their own advantages and limitations. In the next section, a new pruning algorithm which combines the advantages of the previously mentioned algorithms and prunes the network iteratively on a reduced set (if necessary) will be proposed.

III. The Proposed New Pruning Algorithm

In this section, a new pruning algorithm called hybrid sensitivity analysis with re-pruning (HSAR) is investigated. Both weight pruning and node pruning are considered; and most of the disadvantages of the existing algorithms discussed in section two can be eliminated or reduced. Pruning is based on sensitivity calculation and local variance nullity, and the performance of the neural network is revalued using cross-validation at every pruning step. The algorithm iteratively prunes the neural network on a reduced set of connections if a pruning error occurs. The main steps of the algorithm are summarized as follows:

0. Train the oversized neural network using back-propagation and evaluate training performance using cross-validation.

1. Estimate the sensitivity using Eq. (3).

2. Calculate the average sensitivity of each parameter in layer *l* using Eq. (7).

3. Calculate the variance of the sensitivities of the parameters in layer *l* using Eq. (6) and the average sensitivity values calculated from step 2.

4. Find the LPVN of each parameter using Eq. (9) and the average sensitivity values calculated in step 3.

5. Sort the LVPN values from step 4 in an increasing order such that the lowest value is indexed as 0 and the highest value is indexed as *N* (where *N* is the total number of parameters in each step). In case of re-pruning, the parameters with the highest LVPN values are removed from the re-pruning list first (see step 8).

6. Prune any parameter with a LVPN value from step 5 which is less than certain threshold.

7. Train the pruned network and re-gather the sensitivity information.

8. Evaluate the performance of the newly pruned and trained network using the following:

$$\rho(\zeta J'_{TR} + J'_{CV}) < (\zeta J_{TR} + J_{CV}) \tag{10}$$

where ρ is a constant that gives priority to pruned networks, ζ (ζ < 1) is another constant that encourages generalization capacity by favoring the cross validation error over the training error. If the new network fails to show an improvement over the old network, restore the old network to the last working network configuration and go to the re-pruning process in (a); otherwise continue to the next pruning mode (either pruning weight or node).

(a) Choose the reduced parameter list to prune:

$$n_{rp}(t+1) = n_{rp}(t)(1-\lambda) \tag{11}$$

where n_{rp} is the number of weights to re-prune, and λ is the percentage of weights to deduct at each re-pruning step. For node, we have:

$$n_{rp}(t+1) = n_{rp}(t) - 1 \tag{12}$$

i.e., each successive re-pruning step only subtracts one node from the previous attempt. If there are no possible parameters to re-prune, go to the next layer; otherwise continue to (b).

(b) Prune the network using the reduced pruning list from (a).

(c) Back to step 7.

IV. Simulation Results

In this section, the proposed algorithm is applied to solve different pattern classification problems and compared with three existing algorithms described above (i.e, the local sensitivity analysis method (KLSA) [13], the local variance sensitivity analysis (LVSA) [1], and the cross validation pruning algorithm (CVP) [9]). The datasets used in simulation can be found in [4]. They are real-world classification problems related with psychology (e.g., the balance scale data) and medical applications (e.g., the liver disorder and diabetes data sets). They are available from the machine learning repository, University of California, Irvine.

In the following simulation examples, different initial network configurations are considered to fully test the abilities of each pruning algorithm on a variety of hidden node and hidden layer setups. Configuration 1 has one hidden layer with fifteen hidden nodes; configuration 2 has

two hidden layers with ten nodes in each hidden layer; and configuration 3 has three hidden layers with five nodes in each hidden layer. Furthermore, each of the three data sets is divided into ten equal sub-sets, where eight of them are used for training, one is used for validation, and the remaining one is for testing. Each of the sub-dataset is used for training, validation, and testing on a rotation basis, resulting in a total of ten different data configurations.

The weights of all the neural networks are initialized at random before training. The same initial conditions are applied to all the pruning algorithms in each test. To minimize the influences of initial conditions to test results, ten different sets of initial conditions are chosen for each neural network configuration and each sub-dataset. Therefore, for each neural network initial configuration, a total of 10*10 = 100 simulation runs are performed. This process is repeated for each of the four pruning algorithms, and for each of the three data sets.

Table 1 below shows the overall pruning capability of each of the four tested algorithms by displaying the mean and standard deviation (presented as (mean) ± (std)) of the pruning percentage (with respect to the original network). For example, in the balance scale test, for the first neural network configuration, the KLSA algorithm can prune about 6.42% of the total neural network weights (an average for 100 runs with different initial conditions and data rotations), with the standard deviation of 6.20 (also represented in terms of percentage). Similarly, under the same condition, the proposed HSAR algorithm can prune about 24.33% of the total neural network weights, with a standard deviation of 9.05%. Obviously, the new algorithm outperforms the KLSA algorithm.

In tables 2, 3, and 4, the detailed information on the average, standard deviation, the minimum, and the maximum of training error, testing error, number of connections pruned, as well as the pruning percentage for each algorithm on each network configuration is shown. The mean and standard deviation are shown in the form of (mean) ± (std), while the minimum and maximum are shown as [min, max]. For example, in table 2, network configuration 1, for KLSA algorithm, 96.20% of the patterns can be correctly classified (for training dataset), with a standard deviation of 2.03%; the rate for correct classification has a minimum of 90.42% and a maximum of 100%. The performance of the proposed HSAR algorithm is very compatible, with an average rate for correct classification of 95.92% (standard deviation 2.04%), and the minimum and maximum at 87.23% and 99.40%, respectively. However, the big difference on the pruning percentage (average 6.42% for KLSA and 24.33% for HSAR) indicates the neural network pruned by the proposed HSAR algorithm is much smaller.

In summary, the simulation results consistently indicate the HSAR algorithm can reduce the neural network size significantly without sacrificing the network performance.

V. Conclusion

A new hybrid iterative pruning algorithm that can prune multi-layer feedforward artificial neural networks very effectively is presented in this paper. Satisfactory simulation results are obtained, when comparing with three other existing pruning algorithms. More tests will be conducted to further investigate the performance of this new algorithm.

References

[1] Fnaiech, N., Abid, S., Fnaiech, F., and Cheriet, M., "A modified version of a formal pruning algorithm based on local relative variance analysis", First International Symposium on Control, Communications and Signal Processing (2004), pp.849-852

[2] Abid, S., Fnaiech, F., and Najim, M., "A fast Feed-Forward Training Algorithm Using a Modified Form of the Standard Back-Propagation Algorithm", IEEE Transactions on Neural Network (2001), Volume 12, Issue 2, pp.424-430.

[3] Andrews, R., Diederich, J., Golea, M., and Tickle, A.B., "The truth will come to light: Directions and challenges in extracting knowledge embedded within trained artificial neural networks", IEEE Transactions on Neural Networks (1998), Volume 9, Issue 6, pp.1057-1068.

[4] Blake, E. K. C. and Merz, C., "UCI repository of machine learning databases", University of California, Irvine, Department of Information and Computer Science, 1998.

[5] Efe, M.O., Iplikci, S., Kayank, O., and Wilamowski, B., "An Algorithm for Fast Convergence in Training Neural Networks", International Joint Conference on Neural Networks, pp. 1778-1782, Washington DC, July 15-19, 2001.

[6] Engelbrecht, A.P., "A new pruning heuristic based on variance analysis of sensitivity information", IEEE Transactions on Neural Networks (2001), Volume 12, Issue 6, pp.1389-1399.

[7] Giles, C.L. and Lawrence, S., "Overfitting and Neural Networks: Conjugate Gradient and Backpropagation", Proceedings of the IEEE International Conference on Neural Networks (2000), pp.114-119.

[8] Haykin, S., "Neural networks: a comprehensive foundation," Upper Saddle River, N.J., Prentice Hall, 1999.

[9] Huynh, T.Q. and Setiono, R., "Effective neural network pruning using cross-validation", IEEE International Joint Conference on Neural Networks (2005), Volume 2, pp.972-977.

[10] Karnin, E.D., "A simple procedure for pruning back-propagation trained neural networks", IEEE Transactions on Neural Networks (1990), Volume 1, Issue 2, pp.239-242.

[11] Marsland, S., Nehmzow, S.U., and Shapiro, J., "A self-organizing network that grows when required", in Neural Networks, Volume 15 (2002), Isuue 8-9, pp.1041-1058.

[12] Mozer, M. C. and Smolensky, P., "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in Advances in Neural Information Processing (1989) , D.S. Touretzky, Ed., pp.107-115.

[13] Ponnapalli, P.V.S., Ho, K.C., and Thomson, M., "A formal selection and pruning algorithm for feedforward artificial neural network optimization", IEEE Transactions on Neural Networks (1999), Volume 10, Issue 4, pp.964-968.

TABLE I
A Comparison of Each Algorithm's Pruning Percentages

Data	NN	KLSA	LVSA	CVP	HSAR
Balance Scale	1	6.42 ± 6.20	8.63 ± 7.48	9.15 ± 8.16	24.33 ± 9.05
	2	5.66 ± 7.90	9.45 ± 13.31	12.59 ± 14.32	39.61 ± 15.85
	3	2.96 ± 4.33	11.93 ± 9.14	9.44 ± 8.68	33.45 ± 14.91
Liver Disorder	1	6.80 ± 9.96	17.76 ± 21.66	7.90 ± 8.86	32.05 ± 28.58
	2	11.81 ± 11.89	13.13 ± 14.73	20.60 ± 19.79	33.91 ± 18.31
	3	4.14 ± 5.54	18.39 ± 13.72	25.20 ± 21.88	37.99 ± 15.29
Diabetes	1	19.50 ± 22.35	18.14 ± 22.90	15.04 ± 11.77	39.48 ± 31.84
	2	10.13 ± 10.23	19.85 ± 17.19	21.09 ± 22.50	43.42 ± 18.91
	3	4.56 ± 5.65	15.25 ± 12.97	21.53 ± 21.80	45.28 ± 15.03

TABLE II
The Detailed Results for the Balance Scale Data Tests

	KLSA	LVSA	CVP	HSAR
NN config. 1				
Training Accuracy (Percentage)	96.20 ± 2.03	96.31 ± 1.79	96.25 ± 2.08	95.92 ± 2.04
	[90.42 , 100.00]	[92.22 , 99.80]	[90.42 , 100.00]	[87.23 , 99.40]
Testing Accuracy (Percentage)	91.63 ± 3.95	91.16 ± 3.69	91.67 ± 3.67	91.56 ± 3.97
	[83.06 , 100.00]	[79.84 , 99.19]	[83.87 , 100.00]	[82.26 , 100.00]
Percent Pruned	6.42 ± 6.20	8.63 ± 7.48	9.15 ± 8.16	24.33 ± 9.05
	[0.00 , 32.38]	[0.00 , 39.05]	[0.00 , 32.38]	[5.71 , 55.24]
NN config. 2				
Training Accuracy (Percentage)	99.54 ± 0.89	99.71 ± 0.69	96.79 ± 11.91	98.29 ± 2.04
	[94.01 , 100.00]	[96.01 , 100.00]	[41.12 , 100.00]	[90.02 , 100.00]
Testing Accuracy (Percentage)	97.02 ± 1.49	96.98 ± 1.82	94.71 ± 10.56	96.02 ± 2.68
	[93.55 , 100.00]	[90.32 , 100.00]	[34.68 , 100.00]	[84.68 , 99.19]
Percent Pruned	5.66 ± 7.90	9.45 ± 13.31	12.59 ± 14.32	39.61 ± 15.85
	[0.00 , 34.71]	[0.00 , 54.12]	[0.00 , 56.47]	[7.65 , 73.53]
NN config. 3				
Training Accuracy (Percentage)	95.51 ± 2.62	89.93 ± 16.39	96.72 ± 1.90	94.81 ± 3.31
	[88.02 , 99.40]	[38.12 , 99.20]	[87.23 , 99.60]	[81.04 , 99.60]
Testing Accuracy (Percentage)	93.27 ± 4.43	88.02 ± 18.05	93.92 ± 4.16	92.59 ± 4.00
	[78.23 , 100.00]	[13.71 , 100.00]	[78.23 , 100.00]	[79.84 , 99.19]
Percent Pruned	2.96 ± 4.33	11.93 ± 9.14	9.44 ± 8.68	33.45 ± 14.91
	[0.00 , 16.47]	[0.00 , 37.65]	[0.00 , 35.29]	[5.88 , 82.35]

TABLE III
The Detailed Results for the Liver Disorder Data Tests

	KLSA	LVSA	CVP	HSAR
NN config. 1				
Training Accuracy (Percentage)	72.64 ± 5.47	72.08 ± 7.57	73.41 ± 5.63	69.23 ± 9.44
	[58.12 , 84.84]	[48.74 , 84.48]	[55.96 , 83.03]	[39.35 , 84.48]
Testing Accuracy (Percentage)	65.44 ± 7.47	63.69 ± 7.62	64.69 ± 6.82	64.13 ± 7.16
	[45.59 , 80.88]	[39.71 , 79.41]	[48.53 , 79.41]	[42.65 , 79.41]
Percent Pruned	6.80 ± 9.96	17.76 ± 21.66	7.90 ± 8.86	32.05 ± 28.58
	[0.00 , 55.24]	[0.00 , 82.86]	[0.00 , 44.76]	[0.00 , 92.38]
NN config. 2				
Training Accuracy (Percentage)	83.97 ± 3.89	86.19 ± 4.99	78.45 ± 11.11	82.25 ± 6.79
	[75.45 , 92.78]	[68.59 , 96.39]	[44.40 , 94.22]	[62.09 , 95.31]
Testing Accuracy (Percentage)	68.07 ± 6.92	67.60 ± 6.66	68.51 ± 8.88	67.41 ± 6.27
	[42.65 , 82.35]	[47.06 , 82.35]	[30.88 , 80.88]	[47.06 , 82.35]
Percent Pruned	11.81 ± 11.89	13.13 ± 14.73	20.60 ± 19.79	33.91 ± 18.31
	[0.00 , 53.53]	[0.00 , 64.12]	[0.00 , 82.35]	[1.18 , 84.71]
NN config. 3				
Training Accuracy (Percentage)	76.54 ± 3.30	78.19 ± 4.64	67.43 ± 11.24	75.61 ± 6.31
	[68.59 , 83.39]	[55.96 , 86.64]	[41.88 , 83.75]	[53.79 , 84.84]
Testing Accuracy (Percentage)	66.19 ± 5.96	66.94 ± 5.82	64.81 ± 10.92	67.10 ± 6.33
	[41.18 , 77.94]	[50.00 , 77.94]	[25.00 , 79.41]	[45.59 , 79.41]
Percent Pruned	4.14 ± 5.54	18.39 ± 13.72	25.20 ± 21.88	37.99 ± 15.29
	[0.00 , 31.76]	[0.00 , 54.12]	[0.00 , 83.53]	[11.76 , 82.35]

TABLE IV
The Detailed Results for the Diabetes Data Tests

	KLSA	LVSA	CVP	HSAR
NN config. 1				
Training Accuracy (Percentage)	71.78 ± 10.60	70.04 ± 12.52	71.00 ± 11.68	69.20 ± 12.36
	[33.44 , 83.12]	[33.44 , 82.63]	[33.44 , 83.60]	[33.44 , 82.79]
Testing Accuracy (Percentage)	68.61 ± 9.32	65.79 ± 10.39	67.32 ± 9.58	66.68 ± 11.04
	[31.58 , 78.95]	[31.58 , 80.26]	[31.58 , 80.92]	[31.58 , 81.58]
Percent Pruned	19.50 ± 22.35	18.14 ± 22.90	15.04 ± 11.77	39.48 ± 31.84
	[0.00 , 100.00]	[0.00 , 75.56]	[0.00 , 49.63]	[0.00 , 87.41]
NN config. 2				
Training Accuracy (Percentage)	84.66 ± 2.56	86.06 ± 3.19	78.74 ± 14.92	83.40 ± 4.27
	[78.41 , 91.88]	[75.97 , 92.05]	[33.12 , 90.58]	[74.35 , 90.42]
Testing Accuracy (Percentage)	74.63 ± 3.42	74.44 ± 3.94	71.51 ± 13.01	74.70 ± 4.23
	[63.82 , 83.55]	[65.79 , 84.87]	[24.34 , 84.87]	[63.82 , 84.87]
Percent Pruned	10.13 ± 10.23	19.85 ± 17.19	21.09 ± 22.50	43.42 ± 18.91
	[0.00 , 49.47]	[0.00 , 71.05]	[0.00 , 74.21]	[8.95 , 91.05]
NN config. 3				
Training Accuracy (Percentage)	80.79 ± 1.42	81.50 ± 1.99	71.32 ± 17.61	79.89 ± 2.18
	[76.14 , 84.09]	[75.32 , 86.36]	[33.12 , 84.09]	[73.05 , 85.23]
Testing Accuracy (Percentage)	75.75 ± 3.77	75.24 ± 3.50	68.75 ± 15.78	75.93 ± 3.79
	[66.45 , 84.87]	[67.11 , 82.89]	[24.34 , 85.53]	[66.45 , 84.21]
Percent Pruned	4.56 ± 5.65	15.25 ± 12.97	21.53 ± 21.80	45.28 ± 15.03
	[0.00 , 24.21]	[0.00 , 55.79]	[0.00 , 74.74]	[11.58 , 81.05]