On the Optimal Node Ratio between Hidden Layers: A Probabilistic Study

Alan J. Thomas, Simon D. Walters, Saeed Malekshahi Gheytassi, Robert E. Morgan, and Miltos Petridis

***Abstract*—Two-hidden layer feedforward neural networks (TLFNs) have been shown to outperform single-hidden-layer neural networks (SLFNs) for function approximation in many cases. However, their added complexity makes them more difficult to find. Given a constant number of hidden nodes n_h, this paper investigates how their allocation between the first and second hidden layers ($n_h = n_1 + n_2$) affects the likelihood of finding the best generaliser. The experiments were carried out over a total of ten public domain datasets with $n_h = 8$ and 16. The findings were that the heuristic $n_1 = 0.5n_h + 1$ has an average probability of at least 0.85 of finding a network with a generalisation error within 0.18% of the best generaliser. Furthermore, the worst case over all data sets was within 0.23% for $n_h = 8$, and within 0.15% for $n_h = 16$. These findings could be used to reduce the complexity of the search for TLFNs from quadratic to linear, or alternatively for 'topology mapping' between TLFNs and SLFNs, given the same number of hidden nodes, to compare their performance.**

***Index Terms*—ANN, optimal node ratio, topology mapping, two-hidden-layer feedforward, function approximation.**

I. Introduction

Since the introduction of fully interconnected feedforward neural networks into the automotive industry in 1993 [1], they have enjoyed increasing popularity and are still widely used to for function approximation to date, as in for example, [2]. The most popular training algorithm by far is the 'trainlm' algorithm available in the Matlab neural network toolbox. This is an implementation of the Levenberg-Marquardt algorithm [3], which generally yields the best generalisation and fastest convergence for function approximation problems [4]. It is widely used for training both SLFNs [5]-[8] and TLFNs [9], [10], though the former is more common in the literature. Unfortunately it is very memory hungry, and the training time rises exponentially with the number of weights. This makes it unsuitable for networks with more than a few hundred weights [4], and this is particularly problematic for networks with two hidden layers. Because of this, the complexity of an exhaustive search through two hidden layers is actually greater than $O(n^2)$. By way of illustration, whereas an exhaustive search through a single hidden layer with 1 to 32 neurons takes less than an hour, a search through two hidden layers takes approximately 48 hours. This for 30 networks of each topology on a Quad-core i7 4710HQ processor running at 2.5GHz.

This study investigates whether there might be a faster route to take in the search for a TLFN rather than the conventional raster scan. For example only; taking the 'short cut' route shown by the solid line in Fig. 1 would reduce the complexity of the search from quadratic to linear since only 8 topologies are tested instead of 64. However, in order for this method to be useful there must be a high probability that it will find networks with only a small penalty in generalisation error compared to the longer raster route. Furthermore, this should still be the case regardless of the dataset.

In the experiments, ten public domain datasets suitable for function approximation tasks are tested to find a 'short cut' which works well for all. By suitable, it is meant that they have a single output. Although many examples in the literature have multiple outputs (e.g. [9] [10]), this should be avoided if possible since the validation error is calculated across all outputs. This means the training might stop prematurely due to the poor generalisation capability of a single output, perhaps resulting in sub-optimal generalisation of the other outputs. There is also mathematical evidence that reducing the number of outputs increases the storage capacity of the network [11]. If multiple outputs are required, it is better to split the network into several sub-networks each with a single output. These could run in parallel on hardware or multi-threaded software, and are likely to require fewer nodes each, especially for those where the functions are simpler, and/or for those which require fewer inputs.

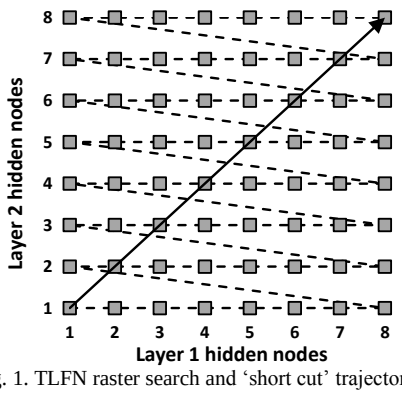

Fig. 1. TLFN raster search and 'short cut' trajectory.

II. Related Work

Two-hidden-layer feedforward neural networks have gained popularity ever since it was shown that they can outperform those with a single hidden layer [12]. Related literature concerning the allocation of nodes between hidden layers can be classified as *mathematical proofs*, *empirical studies* and *anecdotal*. These are discussed in the following

Manuscript received August 5, 2016; revised October 7, 2016.

The authors are with the School of Computing Engineering and Mathematics, University of Brighton, Brighton, BN24GJ UK (e-mail: a.j.thomas@brighton.ac.uk, s.d.walters@brighton.ac.uk, m.s.malekshahi@brighton.ac.uk, r.morgan2@brighton.ac.uk, m.petridis@brighton.ac.uk).

doi: 10.18178/ijmlc.2016.6.5.605

sections:

A. Mathematical Proofs

These are concerned with proving an upper bound for the total number of hidden neurons needed to *exactly* reproduce the training data, or to do so with negligible error. Using Akaike's information criteria, Tamura and Tateshi [13] proved that including the output neuron, the upper bound for SLFNs is N_s, and that for TLFNs this is $N_s / 2 + 3$ where N_s is the total number of training samples. This not only proves that the upper bound for TLFNs is lower than that for SLFNs for a given dataset, but suggests that this would be achieved using $N_s / 2$ neurons in the first hidden layer and two neurons in the second hidden layer.

Inspired by the work in [13], Huang constructed a proof using a network consisting of different neural subnetworks each playing different roles [11]. He rigorously proved that the upper bound on the number of hidden nodes N_h for TLFNs with sigmoid activation function is given by equation (1), where N_o is the number of outputs. These can learn at *any* N_s distinct samples with any degree of precision. Conversely, the storage capacity N_s samples of a TLFN with N_h hidden neurons is *at least* that defined in equation (2).

$$N_h = 2\sqrt{(N_o + 2)N_s} \tag{1}$$

$$N_s = \frac{N_h^{\,2}}{4(N_o + 2)} \tag{2}$$

This proves that reducing the number of outputs increases the storage capacity of a TLFN and that for the best storage capacity, function approximators should be constructed with a single output. Huang further specifies that the total number of hidden neurons N_h should be allocated between the first and second hidden layers (N_1 and N_2 respectively) according to equations (3) and (4).

$$N_1 = \sqrt{(N_o + 2)N_s} + 2\sqrt{N_s/(N_o + 2)} \tag{3}$$

$$N_2 = N_o\sqrt{N_s/(N_o + 2)} \tag{4}$$

For a single output, i.e. N_o=1, equations (1), (3) and (4) can be rewritten as (5), (6) and (7) respectively.

$$N_h = 2\sqrt{3N_s} \tag{5}$$

$$N_1 = \sqrt{3N_s} + 2\sqrt{N_s/3} \tag{6}$$

$$N_2 = \sqrt{N_s/3} = N_h - N_1 \tag{7}$$

Substituting $N_h - N_1$ from (7) into (6), and gathering terms leads to equation (8).

$$3N_1 = \sqrt{3N_s} + 2N_h \tag{8}$$

Finally, by substituting $\sqrt{3N_s}$ for $N_h / 2$ from equation (5) and rearranging, the ratio of nodes between the first hidden layer and the total number of hidden nodes is given by equation (9). So for a single output Huang's proof suggests that a TLFN should have a wide first hidden layer and a narrow second hidden layer with a ratio of 5:1 between them.

$$N_1/N_h = 5/6 \tag{9}$$

However, this ratio relates to the upper bound on the number of nodes, and is merely a consequence of the way that Huang constructed the network for his proof. Another point to consider is that his work is concerned with exactly reproducing the training data (including any noise). So at this upper bound, any network would definitely be overfitting unless a suitable regularisation scheme such as weight decay or early stopping were deployed. The same can also be said for the work of Tamura and Tateshi [13]. This current work differs from both in that it is an empirical study and arrives at a different conclusion for the ten public domain datasets used.

B. Empirical Studies

In a novel study, Thomas et al. [14] compared every possible combination of hidden nodes in the first and second hidden layers for a given constant number of hidden nodes. Mathematically speaking, this is equivalent to all possible $n_1 + n_2 = n_h$, where n_1 and n_2 are the number of hidden nodes in the first and second hidden layers respectively and the total number of hidden nodes (n_h), is constant. This equation describes any of the backward diagonals in Fig. 1. This was repeated for all the different values of constant n_h which is given by the set $n_h = \{34, 20, 16, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3\}$. When the average generalisation errors were plotted centred on the forward diagonal $n_1 = n_2$, (alternatively $n_1 = 0.5n_h$) the result is an '*isonode*' map as shown in Fig. 2. Each solid contour line is referred to as the i^{th} isonode, where $i = n_h$ is the number of hidden nodes. Thus the first contour line in the set, and leftmost on the map is the 34th isonode.

Over two different datasets, it was concluded that the relationship which gives the lowest average generalisation error is given by:

$$n_1 = \text{int}(0.5n_h + 1), n_2 = n_h - n_1 \tag{10}$$

Although the current study can be viewed as an extension of this work in as much as it shares the concept and basic method of data gathering, it is a new study which uses a completely different (probabilistic) approach.

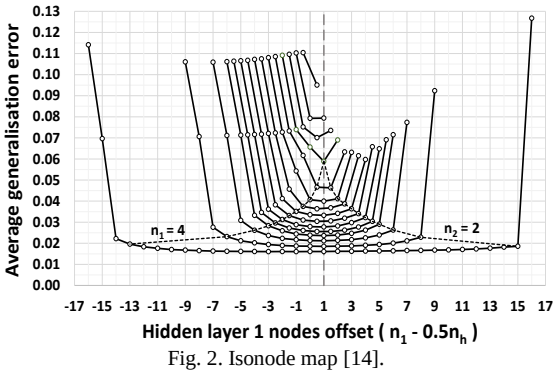

Fig. 2. Isonode map [14].

Unlike the previous study, it uses *relative* generalisation errors which permits direct comparison between node combinations across the entirety of the datasets. This in turn allows probabilistic methods to be used by way of a frequency analysis of the data. Another major difference is five times as many datasets were used in the current study.

C. Anecdotal

In the absence of any other related work, it was decided to

examine what actual node combinations other researchers in the automotive field had selected on for their TLFN designs. This indirect evidence can only be considered anecdotal because in many cases there is no information at all about how these were designed, such as [15] [16]. Others have carried out some form of search but this is unspecified [17], whilst some use trial and error methods [18], [19]. Of those which actually perform a search, some are quadratic [9], [10], whilst other use a linear search with equal number of nodes in the first and second hidden layers [20], [21]. Nonetheless it may be that the authors are basing their searches from past experience. It is impossible to tell.

These choices are summarised in Table I, and the same data represented graphically in Fig. 3. Whatever the underlying reason, or reasoning, it is still interesting that the trend is for an equal number of nodes in the first and second hidden layers, with a slight bias towards more nodes in the first hidden layer.

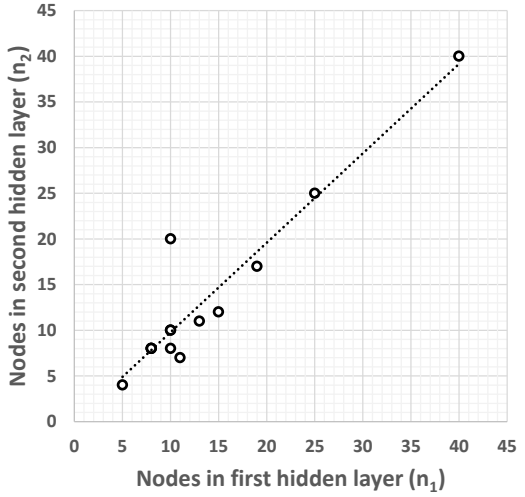

Fig. 3. Graphical representation of node selections.

TABLE I. TLFN NODE SELECTIONS IN THE LITERATURE

Authors	n_1	n_2	n_1 choice
Akcayol and Cinar 2005 [22]	40	40	$0.5n_h + 0$
Hunicz *et al.* 2002 [16]	13	11	$0.5n_h + 1$
Asik *et al*. 1997 [15]	8	8	$0.5n_h + 0$
Czarnigowski *et al.* 2007 [17]	5	4	$0.5n_h + 1$
Deh Kiani *et al*. 2010 [18]	25	25	$0.5n_h + 0$
Janakiraman *et al.* 2006 [23]	10	20	$0.5n_h - 5$
Kurniawan *et al.* 2007 [19]	15	12	$0.5n_h - 1$
Nikzadfar and Shamekhi 2014 [24]	10	10	$0.5n_h + 0$
	10	8	$0.5n_h + 1$
Özgören *et al.* 2013 [25]	11	7	$0.5n_h + 2$
Roy *et al.* 2014 [10]	8	8	$0.5n_h + 0$
Taghavifar *et al.* 2014 [9]	19	17	$0.5n_h + 1$
Wu *et al*. 2004 [20]	8	8	$0.5n_h + 0$
Wu *et al.* 2005 [21]	10	10	$0.5n_h + 0$

III. EXPERIMENTAL METHOD

A. Experimental Environment

All experiments were carried out using Matlab R2014b with the 'fitnet' function available in the neural network toolbox.

B. Datasets

The datasets used in these experiments were selected because of their availability in the public domain, and their suitability for a function approximation tasks. They were obtained from the following sources:

- *UCI Machine learning Repository* [26]*:* Abalone, Airfoil Self-Noise, Concrete Compressive Strength, White Wine.
- *BU Function Approximation Repository* [27]: Kinematics, Mortgage.
- *UP Regression Datasets* [28]*:* Delta elevators.
- *Matlab:* chemical_dataset, engine_dataset, simplefit_dataset. Further information about these can be found by typing 'help' followed by the dataset name at the Matlab command line.

C. Data Preparation

In all cases, the data was split into three subsets: Training, Validation, and Test as summarised in Table II. The Validation set was used to stop the training process when the validation error starts to rise, and the Test set was used exclusively as an estimate of the generalisation error. Prior to this, the Engine dataset, which has two inputs and two outputs was reorganised into three inputs and a single output as in [14].

TABLE II: DATASET SUMMARY

Dataset Name	Inputs	Samples	Training	Val & Test (each)
Abalone	8	4177	3341	418
Airfoil Self-Noise	5	1503	1201	151
Chemical	8	498	398	50
Concrete	8	1030	824	103
Delta Elevators	6	9517	7613	952
Engine	3	1199	959	120
Kinematics	8	8292	6652	820
Mortgage	15	1049	839	105
Simplefit	1	94	64	15
White Wine	11	4898	3918	490

For any given dataset, exactly the same subsets were used for every single network created in the experiments (220,000 per dataset and thus 2.2 million in total). By eliminating any bias in the error surface that may have resulted from a different random split for each network, it was ensured that they were all competing on the same playing field. The only random element at play was thus the initial randomisation of the weights. This initial starting point determines which local minimum in the error surface the training might get stuck in and thus has a direct impact on the generalisation error. For complex error surfaces, it is extremely unlikely that the global minimum will be found.

D. Training Algorithm

In all cases, data preprocessing was 'mapminmax' for both inputs and outputs, the transfer function was 'tansig' and the error function for training was 'mse'. However, the generalisation error in the experiments was reported as the normalised root mean squared error (NRMSE), which is given by:

$$NRMSE_y = \frac{1}{\hat{y}_{max} - \hat{y}_{min}} \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{N_s}} \quad (11)$$

where N_s represents the number of samples, $\hat{y}_i$ is the target value, and y_i is the actual value.

The Levenberg-Marquardt training algorithm 'trainlm' was deployed with early stopping for all experiments, using its default training parameters. Notably: 1000 epochs, training goal of 0, min. gradient of 10^{-7}, and 6 validation failures.

E. Method

It was found in [14] that the difference in generalisation error between adjacent nodes about the axis of symmetry becomes less significant as n_h is increased. This can be seen quite clearly in Fig. 2, where the 34th isonode contour is almost flat. For this reason it was decided to experiment using only two values of n_h, small (n_h = 8) and medium (n_h = 16) for this study. Referring to Fig. 4, for the 8th isonode (n_h = 8), the 7 possible topologies are represented by squares, and the process is as follows:

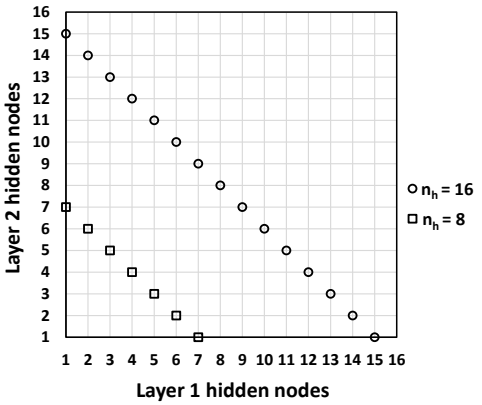

Fig. 4. Experimental topologies for 8th and 16th isonodes.

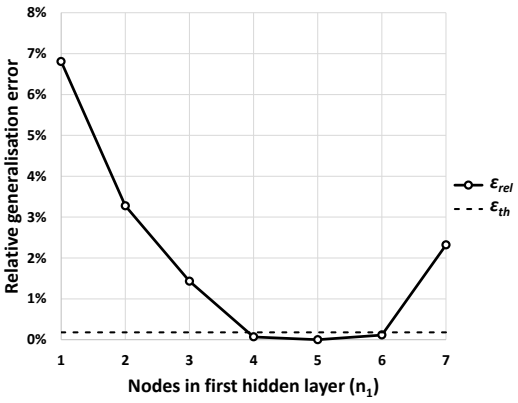

Fig. 5. Raw data for a single run with error threshold ε_{th}.

1. For each topology, find the *median* generalisation error from 100 networks resulting in the set, ε
2. The best generalisation error is $\varepsilon_{min} = min(\varepsilon)$
3. Evaluate the set of relative errors $\varepsilon_{rel} = \varepsilon - \varepsilon_{min}$. This is the raw data for a single run shown in Fig. 5
4. For all topologies such that $\varepsilon_{rel} < \varepsilon_{th}$, register a hit in in its column. In Fig. 5 this is columns 4, 5 and 6
5. Repeat for 100 runs for each of the 10 datasets
6. The probability that a particular topology will yield a generalisation error less than the threshold is given by $p(\varepsilon_{rel} < \varepsilon_{th}) = k / r$, where k is the number of hits for that topology, and r is the total number of runs
7. The results are plotted on a 'probability map', where each column has a probability between 0 and 1.

The process is identical for the 16th isonode (n_h = 16), but in this case the 15 topologies denoted by circles are considered.

IV. Results

The results are presented as the average and worst case probability across the ten datasets for the 8th and 16th isonodes. The error threshold values (ε_{th}) were adjusted, in each case, until the peak worst case probability reached 0.5. At these values, even the worst performing dataset has a 50% chance of being less than the error threshold ε_{th} .This occurred when the value of ε_{th} was 0.18% and 0.13% for the 8th and 16th isonodes respectively. In the probability maps shown in Figs. 6 and 7, the *x-axis* represents the number of nodes in the first hidden layer n_1 (the number of nodes in the second hidden layer $n_2 = n_h - n_1$), and the *y-axis* shows the average and worst case probability that the relative error is less than ε_{th}, or in other words $p(\varepsilon_{rel} < \varepsilon_{th})$.

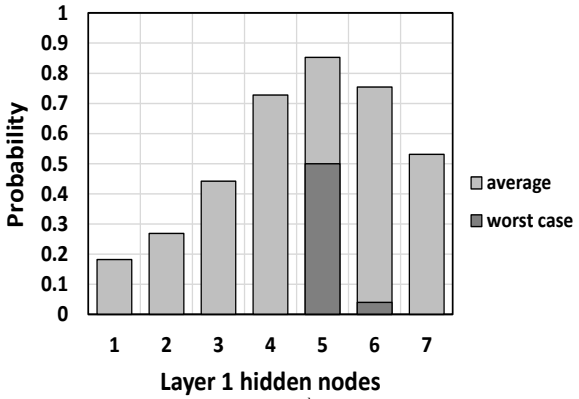

Fig. 6. Probability map for the 8th isonode, with ε_{th} = 0.18%.

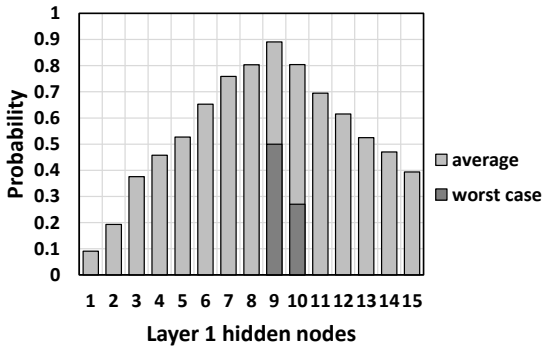

Fig. 7. Probability map for the 16th isonode, with ε_{th} = 0.13%.

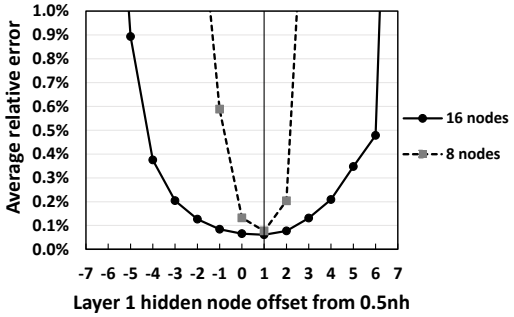

Fig. 8. Average relative error map.

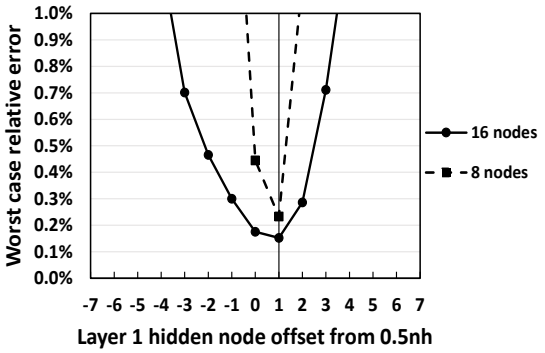

Fig. 9. Worst case relative error map.

Both the average and worst case probabilities peak at n_1 = 5 and n_1 = 9 for the 8th and 16th isonodes respectively. This is consistent with the previous findings of the Authors [14], which concluded that the best ratio of nodes was given by the heuristic $n_1 = int(0.5n_h + 1)$. The probability maps of the individual datasets (not shown here), revealed that some were less susceptible to variations in the node ratio than others, about 50% of them having a fairly flat probability map.

Examining the actual generalisation errors relative to the best generaliser (ε_{rel}), a similar picture emerged. Figs. 8 and 9 show maps of the average and worst case ε_{rel} respectively, against the number of nodes in the first hidden layer. In these maps, the *x-axis* represents an offset from $0.5n_h$ for ease of comparison – in other words $n_1 = 0.5n_h + x$. From these maps it can be seen that on average, at $n_1 = 0.5n_h + 1$, the relative generalisation error ε_{rel} is less than 0.1% for both the 8th and 16th isonodes. Furthermore ε_{rel} is at most 0.15% and 0.23% for the 16th and 8th isonodes respectively.

V. Conclusions

Training feedforward neural networks using backpropagation algorithms is a very probabilistic affair. The initial random allocation of weights dictate the starting position on the error surface, and this in turn dictates the end point. With a perfect training algorithm, the endpoint would always be the global minimum, in which case it wouldn't matter what the initial weights were. Unfortunately, backpropagation algorithms are far from perfect and the endpoint will almost certainly be a local minimum. The consequence of this is that the final generalisation error will vary from training session to training session, and that any search for the 'optimal' topology is more than likely to yield a different outcome on each occasion. In reality, therefore, there is no such thing as an optimal topology, only a probability that a particular topology will yield the optimal generalisation errors.

The same constraints apply to this study, where it is concluded that although the heuristic $n_1 = int(0.5n_h + 1)$ does not guarantee that the best generaliser will be found, there is a high probability that it will – at the very least within the context of the datasets tested here. The findings are that using this heuristic, even the worst performing dataset has a probability of 0.5 of finding a network within 0.18% (n_h = 8) and 0.13% (n_h = 16) of the best generaliser, but that on average these probabilities are much higher at 0.85 and 0.89 respectively. Furthermore, in the *worst* case, the relative errors are within just 0.23% and 0.15% respectively, but typically less than 0.1% in both cases.

These findings are very significant, as they can be applied to reduce the complexity of a quadratic (raster) search through n_1, n_2 to a linear search through n_h with very little penalty in the generalisation error. Such a search would follow the staircase trajectory shown in Fig. 10. Alternatively, only even values of n_h could be considered, in which case the trajectory would be $n_2 = n_1 - 2$, shown as the shaded node allocations in Fig. 10. This latter trajectory would reduce a search through n^2 nodes to one through $n - 2$ nodes. This is equivalent to a reduction in complexity from $O(n^2)$ to $O(n)$.

A further application of these findings is in providing an isonode topology 'transform' or 'mapping' between single and double hidden layer networks. In this context, given a particular number of hidden nodes, n_h, equation (10) is used to map a single hidden layer topology on to an 'equivalent' two hidden layer topology. Fig. 11 shows this topology mapping applied to the Airfoil Self-Noise dataset. It compares the average generalisation error of SLFN and TLFN networks with an equal number of hidden nodes. In this case, node for node, two hidden layers clearly perform better than a single hidden layer beyond n_h = 5.

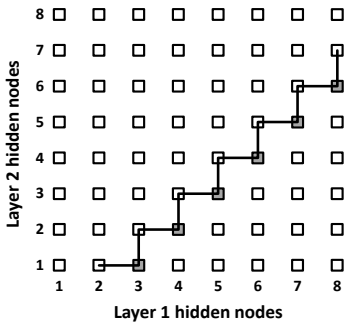

Fig. 10. Proposed search trajectory.

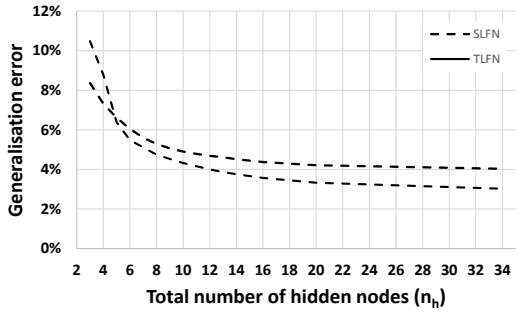

Fig. 11. Isonode topology mapping – Two hidden layers vs. one.

VI. Further Work

It is possible that the complexity could be further reduced to $O(log_2(n))$ using the predictive method developed by the Authors in [29]. There are two possible approaches:

1. Use the predictive method 'as is' to find the optimal SLFN in $O(log_2(n))$. Subsequently, apply an isonode topology mapping to generate an equivalent TLFN.
2. Alternatively, the predictive method could be adapted to use the relationship in equation (10) directly. It was tentatively shown in [14] that the isonode topology mapping could well be suitable for use in such a predictive approach.

It is believed at this stage that the former approach would probably be the faster of the two. However, it is suspected that the latter would yield better results. Pending the results of further experimentation, either approach would reduce the complexity of finding TLFNs from $O(n^2)$ to $O(log_2(n))$. This significant reduction in complexity could potentially find 'optimal' TLFNs in a matter of minutes rather than days.

ACKNOWLEDGMENT

We thank Prof. Martin T. Hagan of Oklahoma State University for kindly donating the Engine dataset used in this paper to Matlab. Thanks also to Prof. I-Cheng Yeh for permission to use his Concrete Compressive Strength dataset [30], as well as the other donors of the various datasets used in this study. A. J. Thomas would also like to thank Chris Thomas for his support.

REFERENCES

[1] M. W. Scaife, S. J. Charlton, and C. Mobley, "A neural network for fault recognition," *SAE International*, Warrendale, PA, SAE Technical Paper 930861, Mar. 1993.
[2] H. Taghavifar, H. Taghavifar, A. Mardani, A. Mohebbi, S. Khalilarya, and S. Jafarmadar, "Appraisal of artificial neural networks to the emission analysis and prediction of CO_2, soot, and NOx of n-heptane fueled engine," *J. Clean. Prod.*, vol. 112, Part 2, pp. 1729–1739, Jan. 2016.
[3] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis*, G. A. Watson, Ed. Springer Berlin Heidelberg, 1978, pp. 105–116.
[4] H. Demuth and M. Beale, "Neural network toolbox user's guide. MATLAB Version 4," 2001.
[5] O. Özener, L. Yüksek, and M. Özkan, "Artificial neural network approach to predicting engine-out emissions and performance parameters of a turbo charged diesel engine," *Therm. Sci.*, vol. 17, no. 1, pp. 153–166, 2013.
[6] A. H. Gadallah, E. A. Elshenawy, A. M. Elzahaby, H. A. El-Salmawy, and A. H. Bawady, "Application of neural networks for prediction and optimization of emissions and performance in a hydrogen fuelled direct injection engine equipped with in cylinder water injection," *SAE International*, Warrendale, PA, 2009-01–2684, Nov. 2009.
[7] B. Xiao, S. Wang, and R. G. Prucka, "A semi-physical artificial neural network for feed forward ignition timing control of multi-fuel SI engines," *SAE International*, Warrendale, PA, 2013-01–0324, Apr. 2013.
[8] J. Rezaei, M. Shahbakhti, B. Bahri, and A. A. Aziz, "Performance prediction of HCCI engines with oxygenated fuels using artificial neural networks," *Appl. Energy*, vol. 138, pp. 460–473, Jan. 2015.
[9] H. Taghavifar, H. Taghavifar, A. Mardani, and A. Mohebbi, "Modeling the impact of in-cylinder combustion parameters of DI engines on soot and NOx emissions at rated EGR levels using ANN approach," *Energy Convers. Manag.*, vol. 87, pp. 1–9, Nov. 2014.
[10] S. Roy, R. Banerjee, A. K. Das, and P. K. Bose, "Development of an ANN based system identification tool to estimate the performance-emission characteristics of a CRDI assisted CNG dual fuel diesel engine," *J. Nat. Gas Sci. Eng.*, vol. 21, pp. 147–158, Nov. 2014.
[11] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 274–281, Mar. 2003.
[12] D. L. Chester, "Why two hidden layers are better than one," in *Proc. International Joint Conference on Neural Networks*, 1990, vol. 1, pp. 265–268.
[13] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layers versus three," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 251–255, Mar. 1997.
[14] A. J. Thomas, S. D. Walters, M. Petridis, S. M. Gheytass, and R. E. Morgan, "Accelerated optimal topology search for two-hidden-layer feedforward neural networks," *Engineering Applications of Neural Networks*, vol. 629, pp. 253–266, Aberdeen: Springer International Publishing, 2016.
[15] J. R. Asik, J. M. Peters, G. M. Meyer, and D. X. Tang, "Transient A/F estimation and control using a neural network," *SAE International*, Warrendale, PA, 970619, Feb. 1997.
[16] J. Hunicz, D. Mazurkiewicz, and A. Niewczas, "Flame spectrum analysis with the use of artificial neural networks," *SAE International*, Warrendale, PA, 2002-01–1145, Mar. 2002.
[17] J. Czarnigowski, M. Wendeker, P. Jakliński, P. Boulet, and F. Breaban, "Idle speed stabilization by neural network model-based control of ignition in SI engine," *SAE International*, Warrendale, PA, 2007-01–2080, Jul. 2007.
[18] M. K. Deh Kiani, B. Ghobadian, T. Tavakoli, A. M. Nikbakht, and G. Najafi, "Application of artificial neural networks for the prediction of performance and exhaust emissions in SI engine using Ethanol-Gasoline blends," *Energy*, vol. 35, no. 1, pp. 65–69, Jan. 2010.
[19] W. H. Kurniawan, S. Abdullah, Z. M. Nopiah, and K. Sopian, "The application of artificial neural network in predicting and optimizing power and emissions in a compressed natural gas direct injection engine," *SAE International*, Warrendale, PA, 2007-01–4264, Oct. 2007.
[20] B. Wu, Z. Filipi, D. Assanis, D. M. Kramer, G. L. Ohl, M. J. Prucka, and E. DiValentin, "Using artificial neural networks for representing the air flow rate through a 2.4 Liter VVT engine," *SAE International*, Warrendale, PA, 2004-01–3054, Oct. 2004.
[21] B. Wu, R. G. Prucka, Z. S. Filipi, D. M. Kramer, and G. L. Ohl, "Cam-phasing optimization using artificial neural networks as surrogate models-maximizing torque output," *SAE International*, Warrendale, PA, 2005-01–3757, Oct. 2005.
[22] A. M. Akcayol and C. Cinar, "Artificial neural network based modeling of heated catalytic converter performance," *Appl. Therm. Eng.*, vol. 25, no. 14–15, pp. 2341–2350, Oct. 2005.
[23] V. M. Janakiraman, S. Suryanarayanan, G. L. N. Rao, and S. Sampath, "Estimation of engine emissions based on physical and chemical properties of biodiesels using artificial neural networks," *SAE International*, Warrendale, PA, 2006-01–3533, Oct. 2006.
[24] K. Nikzadfar and A. H. Shamekhi, "Investigating the relative contribution of operational parameters on performance and emissions of a common-rail diesel engine using neural network," *Fuel*, vol. 125, pp. 116–128, Jun. 2014.
[25] Y. Ö. Özgören, S. Çetinkaya, S. Sarıdemir, A. Çiçek, and F. Kara, "Predictive modeling of performance of a helium charged Stirling engine using an artificial neural network," *Energy Convers. Manag.*, vol. 67, pp. 357–368, Mar. 2013.
[26] M. Lichman, "UCI machine learning repository," University of California, School of Information and Computer Science, 2013.
[27] H. Altay Guvenir and U. Uisal, "BU function approximation repository," Bilkent University, 2000.
[28] L. Torgo. UP regression datasets. University of Porto, Computer Science Department. [Online]. Available: http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html
[29] A. J. Thomas, M. Petridis, S. D. Walters, S. M. Gheytassi, and R. E. Morgan, "On predicting the optimal number of hidden nodes," in *Proc. 2015 International Conference on Computational Science and Computational Intelligence)*, 2015, pp. 565–570.
[30] I.-C. Yeh, "Modeling of strength of high performance concrete using artificial neural networks," *Cem. Concr. Res.*, vol. 28, no. 12, pp. 1797–1808, 1998.

Alan J. Thomas was born in Beirut, Lebanon in 1958. He completed a BSc(Hons) in Electronic Engineering in 1981 and a first class MComp in Computer Science in 2012 at the University of Brighton, UK.

He worked as a cryptographic engineer, senior software engineer, and chief engineer designing the hardware and software for high speed encryption products, embedded systems and gate arrays. He is currently writing his PhD thesis at the University of Brighton, UK, where he is also a part-time lecturer.

He is the author of two peer-reviewed papers: *On Predicting the Optimal Number of Hidden Nodes* (Las Vegas, Nevada: IEEE, 2015) and *Accelerated Optimal Toplogy Search for Two-hidden-layer Feedforward Neural Networks* (Aberdeen, Scotland: Springer, 2016). His research interests include: neural networks, genetic algorithms, neuroevolution, robotics and emissons prediction.

Saeed Malekshahi Gheytassi is associate head of School (Academic affairs) at the School of Computing Engineering & Mathematics, University of Brighton, UK, where he is also a Principal Lecturer in Computer Science.

His research interests include: computer systems architecture, application of artificial intelligence in robotics, and mobile communications.

Miltos Petridis is a professor of Computer Science and the Head of the Knowledge Engineering Group at the University of Brighton. His research and enterprise activities concentrate on applications of Artificial Intelligence, Machine Learning and Case-Based Reasoning on "Big Data" and structurally and semantically complex and uncertain data including Temporal, Spatial, Workflow and Social Networking application data.

Professor Petridis is involved in collaborative research industrial projects in applied Data Mining and Business Intelligence. He is the author of over 100 peer-reviewed publications.

Simon D. Walters was born in Redhill, Surrey, England, in 1968. Between 1990 and 2003 he completed BEng(Hons.) in electrical and electronic engineering, MEng in engineering and business management, a PhD in automotive electronic engineering, and an MBA in business management at the University of Brighton, UK, where he is currently Principal Lecturer.

His previous positions include senior lecturer, senior research fellow, research fellow, electrical engineer and chief test engineer. He has contributed to more than 60 technical publications, and his current research interests include: intelligent systems, high voltage technology, electrical power systems, materials for electrical and electronic systems, and automotive electronics.

Dr. Walters is a chartered engineer (CEng), and a member (MIET) of the Institution of Engineering and Technology. Following his MBA graduation, he was awarded the award for outstanding contribution to the MBA Programme. He is a member of several committees, and reviews technical publications for conferences and journals.

Robert E. Morgan was born in Leeds, UK in 1969. He completed his first degree in Mechanical Engineering at Imperial College London in 1991 and PhD in 1994.

He started career at Ricardo, working in engine research followed by time at Ceres Power and finally Highview Power Storage as Chief Technical Officer. He joined the University of Brighton in 2012 and is currently Assistant Head of the Advanced Engineering Center. He has contributed to over 20 peer-reviewed publications, and is named in 10 patents. His main research interests are on high efficiency engines with waste heat recovery and energy storage. His research is funded by EPSRC, Innovate UK and industry.

Dr Morgan is the current secretary of the UK"s Universities Internal Combustion Engine Group and Director of the Advanced Propulsion Center Internal Combustion Engine Thermal Efficiency spoke.