TOWARDS EVOLUTIONARY DEEP NEURAL NETWORKS

Tomás H. Maul
Computer Science
The University of Nottingham Malaysia Campus
Jalan Broga, 43500 Semenyih,
Malaysia
E-mail: Tomas.Maul@nottingham.edu.my

Andrzej Bargiela
Computer Science
The University of Nottingham
Jubilee Campus, Wollaton Road, Nottingham
UK
E-mail: Andrzej.Bargiela@nottingham.ac.uk

Chong Siang Yew
Computer Science
The University of Nottingham Malaysia Campus
Jalan Broga, 43500 Semenyih,
Malaysia
E-mail: siang-yew.chong@nottingham.edu.my

Abdullahi S. Adamu
Computer Science
The University of Nottingham Malaysia Campus
Jalan Broga, 43500 Semenyih,
Malaysia
E-mail: khyx1asa@nottingham.edu.my

KEYWORDS
Evolutionary artificial neural networks, neuroevolution, deep neural networks, hybrid neural networks.

ABSTRACT

This paper is concerned with the problem of optimizing deep neural networks with diverse transfer functions using evolutionary methods. Standard evolutionary (SEDeeNN) and cooperative coevolutionary methods (CoDeeNN) were applied to three different architectures characterized by different constraints on neural diversity. It was found that (1) SEDeeNN (but not CoDeeNN) changes parameters uniformly across all layers, (2) both evolutionary approaches can exhibit good convergence and generalization properties, and (3) increased neural diversity improves both convergence and generalization. In addition to clarifying the feasibility of evolutionary deep neural networks, we suggests a guiding principle for synergizing evolutionary and error gradient based approaches through layer-change analysis.

INTRODUCTION

This paper attempts to bring together two active but mostly independent research areas within artificial neural networks (ANN), i.e., deep neural networks (DNN) (Hinton et al. 2006) and evolutionary neural networks (ENN) (Yao 1999) (also referred to as neuroevolution). To the best of the authors' knowledge, the literature does not yet show explicit signs of systematic work on evolutionary deep neural networks (EDeeNN). We attempt to take a few early steps in this direction by exploring different evolutionary approaches and different architectural constraints, with the long-term goal of finding efficient and accurate EDeeNNs. We do this in the context of hybrid neural networks (Gutiérrez et al. 2009; Gutiérrez et al. 2011), particularly neural diversity machines (NDM) (Maul 2013). We also investigate how different layers change throughout the optimization process, in order to understand possible underlying causes behind different convergence speeds and generalization capabilities.

The following section gives an overview of the related work, which includes evolutionary and deep neural networks. Subsequently the four main hypotheses underlying the work are summarized, followed by a methodology section which covers neural network architectures, evolutionary algorithms and the experimental setup adopted. After this, the main results are summarized, followed by a discussion and several conclusions.

RELATED WORK

Briefly put, ENN are neural networks whose weights and possibly architectures are tuned primarily via global stochastic optimization algorithms (e.g. genetic algorithms). Early classic references include (Belew et al. 1991) and (Whitley et al. 1990). Recent examples include (Gutiérrez et al. 2011) and (Gauci and Stanley 2010). For useful reviews refer to (Yao 1999) and (Floreano et al. 2008). Although difficulties in scaling ENNs to large models and/or data-sets have prevented this approach from becoming mainstream, ENNs offer several advantages such as the ability to optimize novel types of ANN for which learning algorithms cannot easily be derived and the potential to fulfill the promise of natural computation by effectively incorporating further biological elements into ANN (e.g. evolving developmental and learning rules in modular neural networks).

Deep neural networks (DNNs), which essentially consist of ANN with "many" layers (e.g. four or more connection layers) are not a new concept. They have been studied for many years in models such as the Neocognitron (Fukushima 1980) and Convolutional Neural Networks (LeCun and Bengio 1995). However, it was only after 2006 with the publication of (Hinton et al. 2006), and the development of methods that allowed for the efficient and consistent training of more flexible DNNs, that they finally became a popular topic.

Proceedings 28th European Conference on Modelling and Simulation ©ECMS Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, Marco Castellani (Editors)
ISBN: 978-0-9564944-8-1 / ISBN: 978-0-9564944-9-8 (CD)

Subsequently, many papers have been written and several major machine learning or pattern recognition competitions have been won, based on DNN breakthroughs. For a recent example, refer to the winner of the "Segmentation of neuronal structures in EM stacks challenge - ISBI 2012", which used a DNN with convolutional and max-pooling layers (Ciresan et al. 2012). For a useful review of deep neural networks refer to (Bengio 2009). One of the main reasons why DNNs are so attractive consists of the fact that they consistently form good representations of the underlying causes of the data (automated feature construction) that lead to good generalization properties.

As mentioned, the literature does not yet seem to show any explicit signs of the combination of these two areas. Implicitly EDeeNNs have been studied in NEAT (Stanley and Miikkulainen 2002) where networks can in principle evolve to significant depths, and Neural Diversity Machines (NDM) (Maul 2013), where recurrence can be seen as depth in time. However, explicit studies as reported in this paper still seem to be lacking. More specifically, we address the question of the relative merits of two different evolutionary paradigms: (1) standard evolutionary Deep Neural Networks (SEDeeNN) and (2) cooperative coevolutionary Deep Neural Networks (CoDeeNN), and three different types of architectural constraints: (1) any type of transfer function (TF) for each neuron, (2) any type of TF per layer and (3) any type of TF for the whole network, where a TF consists of a particular combination of a weight function (WF), also commonly referred to as activation function, and a node function (NF), also commonly referred to as output function.

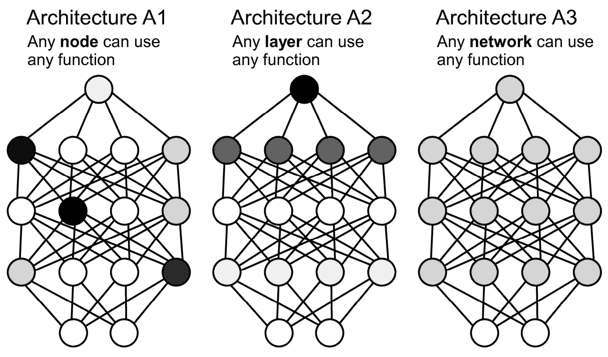
Architecture A1
Any **node** can use any function
Architecture A2
Any **layer** can use any function
Architecture A3
Any **network** can use any function

Figure 1: Three types of neural diversity architectures

HYPOTHESES

The experimental study was guided by four simple hypotheses: (1) in SEDeeNN the amount of change is proportional to layer depth (i.e. deeper (closer to the output) layers change more) whereas in CoDeeNN the amount of change is more balanced across layers, (2) convergence is faster for CoDeeNN compared to SEDeeNN, (3) CoDeeNNs generalize better (have lower test errors on average) than SEDeeNNs and (4) unconstrained architectures converge faster but do not necessarily lead to the best generalization properties. Interestingly our results contradicted most of these hypotheses and revealed further insights and questions, such as the fact that the evolutionary and traditional error gradient approaches appear to be somewhat complementary in terms of how layers change and the question of how best to capitalize on this complementarity.

METHODS

Neural Architectures

The general neural architecture adopted in this paper is based on the NDMs reported in (Maul 2013), which are hybrid ANNs with constraints on the minimal amount of TF diversity allowed. The main difference here is that the connectivity has been restricted to being feedforward, with unrestricted depth or number of layers. This special case of NDMs can be referred to as feedforward NDMs. The main experiment in this paper used networks with four connection layers (i.e. three layers of hidden units), which is minimally but sufficiently deep, where each layer of hidden-units consisted of four nodes.

Several sub-architectures were allowed where each one was characterized by a different degree of freedom with regards to how TF diversity was used. Refer to Figure 1 for a diagrammatic representation of these three architectures. In architecture A1 any TF can be selected for any node, whereas in A2 any TF can be selected for any layer, and in A3 any TF can be selected for the whole network. Thus, only the first two conditions that define an NDM (Maul 2013) are applicable to these restricted (feedforward) NDMs: (1) at least 3 WFs and 3 NFs must be available for nodes and (2) nodes can exhibit any combination of the specified weight and node functions (corresponding to a minimum of 9 TFs).

Table 1 summarizes the weight and node functions used, where a_j refers to the WF output of node j, x_i refers the activation value of inputting node i, w_{ji} refers to the weight of the connection from node i to node j, c refers to some constant and o_j refers to the NF output of node j

Evolutionary Approaches

Two evolutionary approaches were tested, i.e.: (1) standard evolution and (2) cooperative coevolution. In the first case (SEDeeNN), solution vectors were treated as a whole (with no separate treatment for individual layers) and four different global heuristics (variation operators) were employed, i.e.: mutation, cross-over, differential evolution and probabilistic mingling (our implementation of rank-based uniform crossover (Semenkin and Semenkina 2012) and (Ackley 1987)). All heuristics except for the last one are commonly found in the literature. Probabilistic mingling is a form of cross-over where a pair of solutions (e.g. solutions s1 and s2) is scanned parameter by parameter, in order to create a third solution (e.g. solution s3). If s1 is fitter than s2 then the probability of selecting a parameter from s1 (for inclusion in s3) is 0.75 as opposed to 0.25 for s2, and conversely if s2 is fitter. Population dynamics was based on initialization, expansion of the initial set using global heuristics, trimming by an elitist and diversity-preserving selection method and possible padding with random solutions (refer to Algorithms 1 and 2 for high-level overviews of the different optimization approaches).

Table 1: NDM weight and node functions

Type	Functions	Equations
WF	Inner product	$a_j = b_j + \sum_{i=1}^{n} w_{ji} x_i$
WF	Euclidean distance	$a_j = \sqrt{\sum_{i=1}^{n} (w_{ji} - x_i)^2}$
WF	Product	$a_j = \prod_{i=1}^{n} c w_{ji} x_i$
WF	Higher-order product	$a_j = \prod_{i=1}^{n} \left\| x_i^{w_{ji}} \right\|$
WF	Standard deviation	$a_j = \left(\sum_{i=1}^{n} w_{ji}\right) \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2}$
WF	Minimum	$a_j = \min(w_{j1} x_1 \cdots w_{jn} x_n)$
WF	Maximum	$a_j = \max(w_{j1} x_1 \cdots w_{jn} x_n)$
NF	Identity	$o_j = a_j$
NF	Sigmoid	$o_j = \frac{1}{1 + e^{-c a_j}}$
NF	Gauss. w/ thresh.	$o_j = e^{-\frac{(a_j)^2}{c}}$, if $o_j = d$ then $o_j = 1$

For the basic CoDeeNN, we reused as many aspects of SEDeeNN as possible (e.g. global heuristics and selection method) to facilitate comparisons. The main difference pertained to the creation of multiple sub-populations, one for each layer. The main challenge here consisted of the evaluation of each sub-solution in each sub-population. Note that each sub-solution represents a single layer and therefore cannot be evaluated independently. Evaluation was done in a similar way as with the Enforced SubPopulations method (Gomez and Miikkulainen 2003), i.e.: a sufficient number of whole solution vectors was created from random combinations of sub-solutions (one for each layer sub-population), which could then be evaluated in the normal way, and whereby the cost of a sub-solution could be subsequently computed by averaging the cost of all the networks it participated in.

Experimental Setup

The paper's main experiment was designed to verify the four hypotheses outlined in the Hypotheses section. This experiment involved running 30 tests for every possible combination of architecture (i.e. A1, A2 and A3) and evolutionary approach (i.e. SEDeeNN and CoDeeNN), and storing information on training and test errors across 100 generations. The experiment was conducted using five different data-sets, i.e.: XOR, bar-circle, u-shape, Iris and Accute Inflammations (hereafter abbreviated as Inflammation). The bar-circle and u-shape datasets are two simple and synthetic data-sets (depicted in Figure 2), whereas the last two datasets were obtained from the UCI Machine Learning Repository (Blake and Merz 1998). Apart from training and test errors, two other performance metrics were computed, i.e.: relative generalization capacity (RGC) and average layer-change. The former metric was computed for each combination of data-set, evolutionary approach and architecture, by dividing average training errors by average test errors for each generation, and then averaging all of these ratios. The latter was determined by computing the average parametric absolute change for each layer of the best solution (across 30 tests), generation by generation, and then averaging these changes (across 100 generations). For more information relating to algorithms and parameter settings please contact the corresponding author.

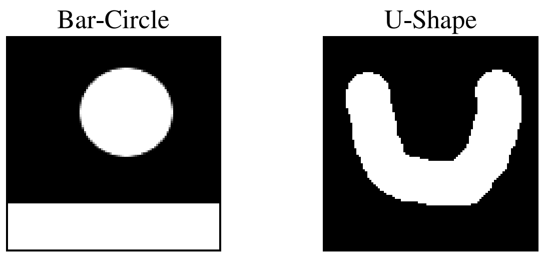

Figure 2. Example synthetic data-sets

RESULTS

Table 2 gives an overall summary of the results. The first column depicts for each data-set, a list of approaches ordered from best to worst in terms of average training error. This column is indicative of the convergence properties of different approaches. Several simple observations are possible here: (1) results for experiments involving architectures A1 and/or A2 consistently lead to faster convergence (seven cases for A1 and three cases for A2), (2) both SEDeeNN and CoDeeNN led to fastest convergences (in bold), (3) CA3 was consistently the worst (slowest) approach, (4) CoDeeNN was used in the best approaches for the more complex data-sets (i.e. Iris and Inflammation) whereas SEDeeNN was used in the best approaches for the simpler data-sets (i.e. XOR, bar-circle and u-shape), (5) the A2 architecture led to the fastest convergence for two data-sets (i.e. u-shape and Inflammation). Refer to the left-hand side of Figure 3 for the averaged training curves (over 30 tests) for the bar-circle, u-shape, Iris and Inflammation data-sets. The x-axes depict generations, whereas the y-axes depict classification errors. These results demonstrate the usefulness of neural diversity in evolutionary deep neural networks, both in terms of convergence and generalization. Refer to Figure 4 for training and test error curves averaged across data-sets and architectures.

It is important to note that although CoDeeNN converged the fastest for two cases (i.e. Iris and Inflammation), this convergence was measured in terms of training error relative to generation and not actual computational time. In actuality, because of the difficulty and computational complexity of evaluating layers in a cooperative coevolutionary context, CoDeeNN is significantly slower than SEDeeNN.

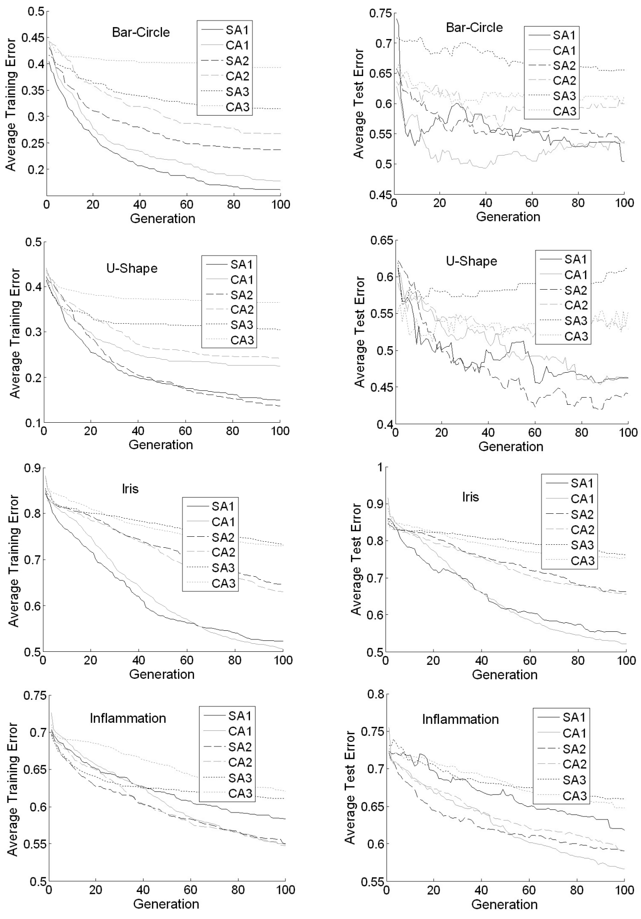

Figure 3. Training errors (left) and test errors (right)

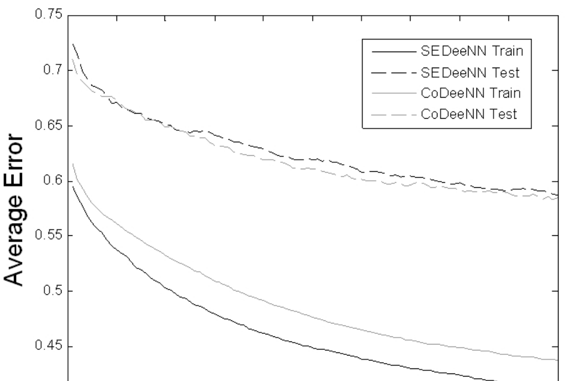

Figure 4: Average errors across bar-circle, u-shape, Iris and Inflammation

The second column in Table 2 is similar to the first column except that the ordering of approaches is done according to the average test error at generation 100. This column is indicative of the generalization properties of different approaches. Because of the small size and nature of the XOR data-set we do not use it here to draw conclusions regarding generalization. As before, several simple observations can be made: (1) the generalization ranking of approaches (second column) is similar (although not equivalent) to the convergence ranking of approaches (first column), (2) the top-two generalizers always involve architectures A1 and/or A2 (six cases for A1 and two cases for A2), (3) the best generalizers may involve either SEDeeNN or CoDeeNN, (4) the worst generalizers always involve architecture A3 (as mentioned we exclude XOR), (5) the two worst generalizers were always CA3 and SA3 and the latter was consistently the worst one. The right-hand side of Figure 3 depicts average test error curves for the bar-circle, u-shape, Iris and Inflammation data-sets.

Algorithm 1: High-level overview of SEDeeNN

```
1:  function SEDeeNN(data, p)
2:      noStop = true;
3:      trimSols = initializeSolutions(data, p);
4:      While noStop
5:          coSols = doCrossOver(trimSols, p);
6:          pmSols = doProbabilisticMingling(trimSols, p);
7:          mSols = doMutation(trimSols, p);
8:          deSols = doDifferentialEvolution(trimSols, p);
9:          sols = [trimSols; coSols; pmSols; mSols; deSols];
10:         sols = evaluateSolutions(sols, data, p);
11:         sols = sortSolutions(sols);
12:         trimSols = trim(sols, p);
13:         noStop = check stopping conditions;
14:     End
```

Algorithm 2: High-level overview of CoDeeNN

```
1:  function CoDeeNN(data, p)
2:      noStop = true;
3:      pops = initializePopulations(p);
4:      While noStop
5:          pops = evaluateSolutionsCoCo(pops, data, p);
6:          For i = 1 to number of populations
7:              aPop = pops{i};
8:              coSols = doCrossOver(aPop, p);
9:              pmSols = doProbabilisticMingling(aPop, p);
10:             mSols = doMutation(aPop, p);
11:             deSols = doDifferentialEvolution(aPop, p);
12:             pops{i} = [coSols; pmSols; mSols; deSols];
13:         End
14:         noStop = check stopping conditions;
15:     End
16:     pops = evaluateSolutionsCoCo(pops, data, p);
```

The third column in Table 2 depicts those approaches which obtained the largest relative generalization capacity. The fact that CA3 obtained the best RGC score in three cases is most probably more a reflection of the fact that it was the poorest converger. However, what is more noteworthy is that for the Inflammation data-set, CA1 obtained the highest RGC score when it was also the best generalizer (second column). In the very least this provides more evidence that increased neural diversity (i.e. A1) does not necessarily impact generalization capacity in a negative way, either in an absolute (i.e. average final test error) or relative (i.e. RGC) sense.

Columns four and five address the question of whether different layers change on average to different extents for different approaches. The most obvious observation here is that for CoDeeNN there is always differential layer-change, regardless of the architecture and data-set used, whereas for SEDeeNN, in an overwhelming majority of cases, layer-change is uniform (the two possible exceptions are "bar-circle SA3" and "Iris SA2"). Refer to Figure 5 for specific examples of the distinction between SEDeeNN and CoDeeNN in terms of layer-changes, using different data-sets. The y-axes depict the average amount of parametric change per layer, normalized by the number of parameters used by each layer. These examples show how SEDeeNN tends to be associated with uniform layer-changes, whereas CoDeeNN, where layers evolve semi-autonomously, is associated with different degrees of parametric change per layer. Legend: conL1 … conL2 = connection layer 1 … connection layer 4.

Considering the Inflammation case (Figure 5, bottom row) across architectures A1, A2 and A3, notice how the middle layers always exhibit maximal layer-change, whereas the last layer always exhibits minimal change, and the first layer varies. Furthermore notice how architecture A1 involves the least overall change. This is in contrast to layer-change dynamics in error gradient based approaches, where it is generally hard for error-based information to penetrate earlier layers, and therefore suggests a useful guiding principle for the integration of evolutionary and gradient based approaches.

Several sanity checks were also conducted using SEDeeNN with an A1 architecture adopting seven connection layers (node/layer architecture: [2 8 4 4 4 4 4 1]), on several simple data-sets including XOR and other 2D data-sets similar to those in Figure 2, whereby it was demonstrated that deeper networks could indeed converge to zero training error in less than 100 generations.

DISCUSSION

Hypotheses

The paper's first hypothesis was completely disproven by the experimental results, which demonstrated that layer-change tends to be more consistent across layers for SEDeeNN, in contrast to CoDeeNN where the middle layers tend to change the most and the last layer tends to change the least (see Figure 5). Looking back at Table 2, one can see that 15 out of 15 CoDeeNN cases involve differential layer-change, whereas only 2 out of 15 SEDeeNN cases involve differential layer change. The table legend consists of: Train (100) = list of approaches, ordered from best to worst, in terms of average training error at generation 100; Test (100) = list of approaches, ordered from best to worst, in terms of average test error at generation 100; SA1 … CA3 = SEDeeNN Architecture 1 … CoDeeNN Architecture 3; Max RGC = approach with largest relative generalization capacity; LC = layer-changes; s = no significant difference between layers in terms of layer-changes; D = significant difference between layers in terms of layer-changes.

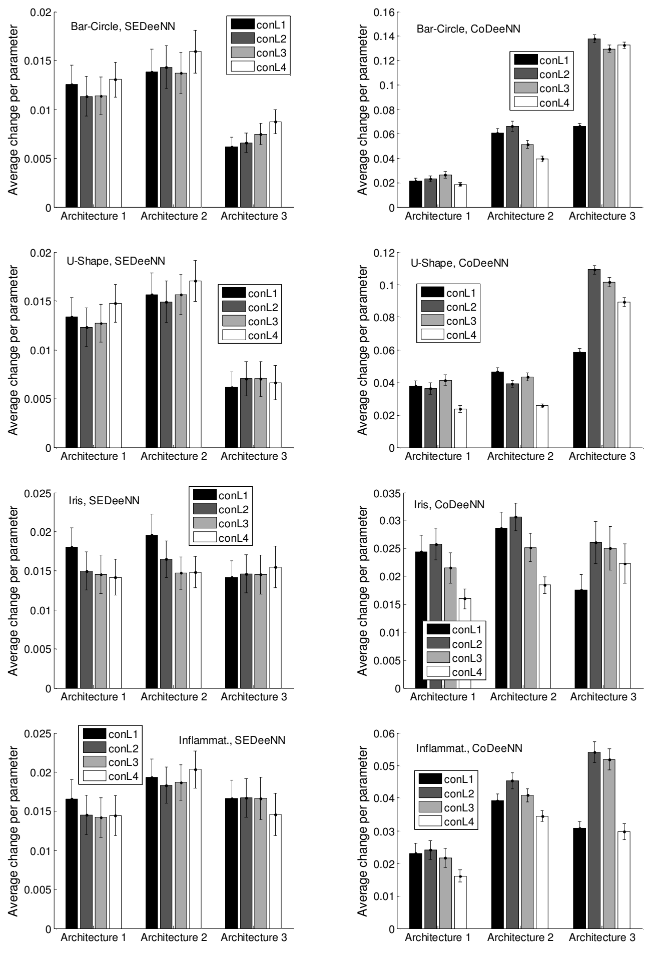

Figure 5: Layer changes for SEDeeNN and CoDeeNN

The second hypothesis was also disproven. Overall SEDeeNN seems to have better convergence properties (e.g. XOR, bar-circle and u-shape) although there are cases where CoDeeNN seems to be better (e.g. Iris and Inflammation). Going back to Table 2, it can be seen that 6 out of the 10 top-two (winner and runner-up) convergers for all data-sets use SEDeeNN whereas 4 out of 10 use CoDeeNN. Looking at training error curves averaged across datasets and architectures (Figure 4) the overall convergence advantage of SEDeeNN is clear. Note that this conclusion, so far, is only valid for a basic implementation of cooperative coevolution of layers in a deep neural network. It is possible that more advanced implementations may change this conclusion.

The third hypothesis was also disproven. Although there is definite evidence for the generalization capability of CoDeeNN (e.g. Inflammation and Iris), SEDeeNN is by no means inferior (e.g. bar-circle and u-shape). In fact, closer inspection of Table 2 reveals that 5 of the 8 top-two generalizers across all data-sets (excluding XOR) use SEDeeNN whereas only 3 out 8 use CoDeeNN. Notice also how SA2 has the highest RGC score for Iris and how the SEDeeNN and CoDeeNN average test error curves in Figure 4 are virtually indistinguishable.

Table 2: Summary of results

Data	Train (100)	Test (100)	Max RGC	SEDeeNN LC			CoDeeNN LC		
				A1	A2	A3	A1	A2	A3
XOR	**SA1**, SA2, SA3, CA1, CA2, CA3	**SA2**, SA3, SA1, CA3, CA2, CA1	CA3	S	S	S	**D**	**D**	**D**
Bar-circle	**SA1**, CA1, SA2, CA2, SA3, CA3	**SA1**, CA1, SA2, CA2, CA3, SA3	CA3	S	S	**D**	**D**	**D**	**D**
U-shape	**SA2**, SA1, CA1, CA2, SA3, CA3	**SA2**, SA1, CA1, CA2, CA3, SA3	CA3	S	S	S	**D**	**D**	**D**
Iris	**CA1**, SA1, CA2, SA2, CA3, SA3	**CA1**, SA1, CA2, SA2, CA3, SA3	SA2	S	**D**	S	**D**	**D**	**D**
Inflammat.	**CA2**, CA1, SA2, SA1, SA3, CA3	**CA1**, SA2, CA2, SA1, CA3, SA3	CA1	S	S	S	**D**	**D**	**D**

The first part of the fourth and final hypothesis was confirmed. In general the more unconstrained the architecture the faster the convergence. A1 is the most unconstrained architecture and tends to be the most highly ranked one (average convergence ranking; not shown). Conversely A3 is the most constrained architecture and tends to be the lowest ranked one. Note that the u-shape and Inflammation data-sets provide interesting exceptions in that A2 converges faster than A1. Note how in Table 2, the second half of the hypothesis was mostly disproven, since in spite of unconstrained diversity allowing greater convergence speed this did not consistently affect generalization in a negative way. Unconstrained architectures not only tended to provide better convergence but also better generalization properties.

Neural Diversity

The fact that these results mostly disprove the original hypotheses is good news for neural diversity. Neural diversity was repeatedly shown to improve convergence speed without simultaneously jeopardizing generalization capacity. Note that the more unconstrained an architecture is, the more neural diversity it exhibits. The results show that neural diversity can be added to deep neural networks, leading to improved convergence and generalization. Moreover, this addition doesn't even demand a more sophisticated optimization approach such as cooperative coevolution in order to deal with the typical issues posed by deep layers. On the contrary, a standard evolutionary approach characterized by basic global heuristics such as mutation, cross-over, differential evolution and probabilistic mingling, was shown to be sufficiently effective.

Synergizing Evolution and Error Gradients

In general, one of the major problems with deep neural networks pertains to vanishing error gradients, which makes it difficult for error derivatives to percolate from outer to inner layers. In EDeeNN this is indirectly reflected in how much different layers change from generation to generation (this is the ENN version of the problem). Before running the experiments it was believed that a standard evolutionary approach would lead to more extensive changes at outer layers, with limited changes at inner layers, mirroring the error gradient case. Fortunately the results contradicted this expectation, demonstrating that SEDeeNN involves uniform layer changes whereas CoDeeNN involves relatively more changes in middle and inner layers. This suggests that evolutionary and error gradient based approaches can be complementary, and that the potential benefit of this synergy is likely to be more fully exploited if we are guided by a deeper understanding of layer-change within each approach and in the context of their integration.

Conclusions

This paper has shown that neural diversity tends to improve the convergence and generalization properties of small deep neural networks. Moreover, standard evolutionary algorithms appear to be at least as good as cooperative coevolution in optimizing deep neural networks with diverse transfer functions. Also, standard evolutionary methods were shown to be capable of changing parameters consistently across all layers. We believe that the significance of this work lies in the demonstration that neural diversity, standard evolutionary methods and the analysis of layer-changes are all fruitful priorities for future work in the area of evolutionary deep neural networks.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for helpful comments and the Faculty of Science, University of Nottingham Malaysia Campus, for its support.

REFERENCES

Hinton, G.E., Osindero, S. and Teh, Y.W. 2006. "A fast learning algorithm for deep belief nets," Neural computation, 18(7), pp. 1527-1554.

Yao, X. 1999. "Evolving artificial neural networks," Proceedings of the IEEE, 87(9), 1423-1447.

Gutiérrez, P.A., Hervás, C., Carbonero, M. and Fernández, J.C. 2009. "Combined projection and kernel basis functions for classification in evolutionary neural networks," Neurocomputing, 72(13), 2731-2742.

Gutiérrez, P.A. and Hervás-Martínez, C. 2011. "Hybrid artificial neural networks: models, algorithms and data". In Advances in Computational Intelligence, Springer Berlin Heidelberg, 177-184.

Maul, T. 2013. "Early experiments with neural diversity machines," Neurocomputing, 113, 36-48.

Belew, R.K., McInerney, J. and Schraudolph, N.N. 1991 "Evolving networks: Using the genetic algorithm with connectionist learning," In Proc. Second Conference on Artificial Life, 511-547.

Whitley, D., Starkweather, T. and Bogart, C. 1990. "Genetic algorithms and neural networks: Optimizing connections and connectivity," Parallel computing, 14(3), 347-361.

Gutiérrez, P.A., Hervás-Martínez, C. and Martínez-Estudillo, F.J. 2011. "Logistic regression by means of evolutionary radial basis function neural networks," IEEE Transactions on Neural Networks, 22(2), 246-263.

Gauci, J. and Stanley, K.O. 2010. "Autonomous evolution of topographic regularities in artificial neural networks," Neural computation, 22(7), 1860-1898.

Floreano, D., Dürr, P. and Mattiussi, C. 2008. "Neuroevolution: from architectures to learning," Evolutionary Intelligence, 1(1), 47-62.

Fukushima, K. 1980. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biological Cybernetics, 36(4), 93-202.

LeCun, Y. and Bengio, Y. 1995. "Convolutional networks for images, speech, and time series," In The handbook of brain theory and neural networks, Cambridge, MA: MIT Press.

Ciresan, D., Giusti, A. and Schmidhuber, J. 2012. "Deep neural networks segment neuronal membranes in electron microscopy images," In Advances in Neural Information Processing Systems 25, 2852-2860.

Bengio, Y. 2009. "Learning deep architectures for AI," Foundations and trends in Machine Learning, 2(1), 1-127.

Stanley, K.O. and Miikkulainen, R. 2002. "Evolving neural networks through augmenting topologies," Evolutionary computation, 10(2), 99-127.

Gomez, F.J. and Miikkulainen, R. 2003. "Active guidance for a finless rocket using neuroevolution," In Genetic and Evolutionary Computation—GECCO, 2084-2095.

Blake, C. and Merz, C.J. 1998. "UCI Repository of machine learning databases."

Semenkin, E., and Semenkina, M. 2012. "Self-configuring genetic programming algorithm with modified uniform crossover". In IEEE Congress on Evolutionary Computation (CEC), pp. 1-6.

Ackley, D. 1987. "A Connectionist Machine for Genetic Hillclimbing", vol. 28 of The Kluwer International Series in Engineering and Computer Science, Kluwer Academic, Boston, Mass, USA, 1987.

AUTHOR BIOGRAPHIES

TOMAS H. MAUL was born in Madeira, Portugal and did a BSc. in Biological Psychology at the University of St. Andrews, an MSc. in Computer Science at Imperial College and a PhD. in Computational Neuroscience at the University of Malaya. He worked for two years at MIMOS Bhd. as a Senior Researcher in the fields of Pattern Recognition and Computer Vision. He is currently an Associate Professor at the University of Nottingham Malaysia Campus, where he conducts research in the areas of Neural Computation, Optimization and Computer Vision. His e-mail address is Tomas.Maul@nottingham.edu.my and his Web-page is `http://kcztm.jupiter.nottingham.edu.my/`

ANDRZEJ BARGIELA is Professor of Computer Science at the University of Nottingham. Until recently he was Director of Computer Science at the University of Nottingham, Malaysia Campus. He is a member of the Automated Scheduling and Planning research group in the School of Computer Science at the University of Nottingham. Since 1978 he has pursued research focused on processing of uncertainty in the context of modelling and simulation of various physical and engineering systems.

CHONG SIANG YEW received the B.Eng. (Hons.) and M.Eng.Sc. degrees in electronics engineering from Multimedia University, Melaka, Malaysia, in 2002 and 2004, respectively, and the Ph.D. degree in computer science from the University of Birmingham, Edgbaston, Birmingham, U.K., in 2007. He is currently an Honorary Research Fellow with the School of Computer Science, University of Birmingham. He was a Research Associate with the Centre of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, in 2007. He joined the School of Computer Science, University of Nottingham, Semenyih, Malaysia, in 2008. He is currently a member of the Automated Scheduling, Optimization and Planning Research Group, School of Computer Science, University of Nottingham, U.K.

ABDULLAHI SHUAIBU ADAMU received the B.S.(Hons) degree in Computer Science from the University of Nottingham Malaysia Campus, Malaysia, in 2011 and is currently a PhD candidate at the same University within the School of Computer Science. Abdullahi Adamu is conducting research in the area of hybrid artificial neural networks, with special focus on the issues and opportunities surrounding neuronal diversity, in the context of neural diversity machines.