Evolving Deep Neural Networks : A New Prospect

Conference Paper · August 2016

DOI: 10.1109/FSKD.2016.7603153

3 authors:

Sreenivas Sremath Tirumala
Auckland University of Technology

Shahid Ali
Virtual University of Pakistan

Phani Ramesh
Sri Venkateswara University

Evolving Deep Neural Networks : A New Prospect

Sreenivas Sremath Tirumala
Auckland University of Technology,
Auckland, New Zealand

Ali S
UNITEC Institute of Technology,
Auckland, New Zealand

C Phani Ramesh
Sri Venkateswara University,
Tirupati, India

***Abstract*—The success of Deep Neural Networks (DNNs) for various applications like Natural language processing (NLP), image processing, hand-written character recognition inspired to use other machine learning (ML) paradigms like Support Vector Machines (SVM), Reinforcement Learning (RL) and Evolutionary Computation (EC) techniques for improving learning process. Using evolutionary algorithms to improve the efficiency of deep learning attained some success. However, these techniques are unable to reduce the learning time which is the key concern for deep learning. The main problem with DNN is that, it uses a random topology to start with (similar to artificial neural networks). If the topology is not suitable, training procedure will restart with a new topology and this process continues till expected results are obtained.**

In this paper, we propose, for the first time, a new prospect for evolving optimized deep neural networks which can provide a warm start to the training process compared to heuristic random initial architecture. We discuss the theoretical approach towards possibility of optimizing the learning process inspired from the existing un-conventional approaches. The training process of DNN with EC approach is faster than regular approach by a considerable difference of over 6 hours for MNIST data set. Further, we observed a considerable improvement in the classification accuracies. Our approaches resulted in an improved classification accuracy of 2% and 4.4%for MNIST data set and 1.2% and 1.4% for IRIS data set compared to heuristic random weights approach. Our initial experimental results prove that evolutionary approaches provides a warm start to the deep learning, thus, reducing the training time.

Keywords: Evolving Neural Networks, Deep Neural Networks, Multi-population evolution, Co-evolution

I. Introduction

Deep Learning has attained focus since 2006 when for the first time, an efficient procedure to train deep architectures has been proposed by G.E. Hinton [1]. Deep learning showed considerable success in Nature language processing, image recognition and other patter recognition implementations [2] [3]. Deep learning is a form of hierarchical learning whose theoretical concepts were first proposed by Lecun in 1998 [4]. Though Lecun's Convolutional Neural Networks (ConvNets or CNNs) have produced good results, the training procedure was very time consuming. In 2006, Lecun, Hinton and Bengio proposed a layers wise training procedure for three different implementations of deep architectures and attained state of art results for various problems. Recent implementation of neural networks for gene expression data also attained good results [5].

The entire deep learning implementation can be divided into two parts, initial design of architecture and training procedure. Initially, a random topology is constructed and is trained using unsupervised training. In the second stage a layer-wise training procedure is adopted and finally the entire network is trained using supervised training mechanism. Deep learning is considered as a slow process as it is time consuming to optimize a randomly selected topology. DNN uses gradient descent in layer-wise training increases the chances of falling in local optima which results in in-efficient and undesirable learning procedure.

Artificial Neural Networks (ANNs) faced similar issue that was resolved by applying Evolutionary Computation (EC) techniques. By EC algorithms, optimized neural networks are evolved from simple ANNs and the process is termed as Neuroevolution (NE) [6]. NE algorithms like NEAT has shown considerable success in various real world applications. Further NE eliminates the problem of vanishing error gradient which affects the learning of ANNs [7]. In-line with the success of NE, we try to analyse the possibility of evolving deep neural networks and optimizing them using EC principles to improve the deep learning process.

In this paper, we propose a new prospect of evolving optimized DNNs which can give a warm start to the procedure of deep learning and reduce the amount of time for training. Our initial experimental produced promising results. This paper is presented as follows. Section II consists of previous work on Neuroevolution and deep learning along with related work on optimising deep learning process. The proposed approach is presented in Section III. Experimental results is presented in Section IV followed by Conclusion and Future work as Section V.

II. Related work

A. Neuroevolution

Evolutionary Computation (EC) refers to general computational methods inspired by Darwin's principles of natural evolution for problem solving [8]. In practice, a computer algorithm that implements the principles of biological evolution and the natural selection process is commonly known as an Evolutionary Algorithm (EA). EA has been widely used in areas that involve combinatorial optimization and is effective in identifying efficient solutions for many real world optimization problems. ANNs are used to represent and solve complex problems due to their ability to learn, flexibility with network topology for required output [9]. With ANNs, the required output can be achieved either by adjusting weights or and network topology. ANNs have high success rate as functional approximators for policy search algorithms like EA and Temporal-Difference methods to represent solutions for Learning problems [10]. NE is the process of implementing EAs for designing and training ANNs [6] and the ANNs thus evolved are called Evolutionary Artificial Neural Networks (EANNs). The NE algorithms can distinguished depending on what exactly is evolved, network topology or connection weights or both (Topology and Weight Evolving Artificial Neural Networks - TWEANN). Further NE can be used to generate problem based optimized network topologies to measure the performance where there is no standard format of fixed input and output.

NE uses Genetic Algorithm (GA) to search ANN policies for obtaining a most fit individual among a population of solutions [6]. Since ANNs are universal approximators, this mapping leads to a powerful binding. The recurrent neural networks that have feedback connections can maintain their internal state obtained from previous observed inputs. With this ANNs can solve partially observable tasks. This scenario follows evolving efficient ANNs instead of training ANNs to be efficient. With this, NE eliminates the problem of vanishing error gradient which affects the learning of ANNs [7]

There are two major approaches to evolving ANN architectures. One is the evolution of pure architectures (i.e. architectures without weights) and connection weights will be trained after a near optimal architecture has been found. The other approach is simultaneous evolution of both topologies and weights [11] [6],. When both architecture and weight information is encoded into individuals, the impact of random initial weights and training algorithm will be considerably reduced [12] [13]. But, the possibilities of evolving similar networks with hidden units defined in different orders (very dissimilar combinations), this may prevent successful recombination which is referred as permutation or competing conventions problem [12]. However, this problem can be resolved by the evolutionary programming based approaches [14] [15].

There are many NE implementations based on the type of the problem, type of encoding and EC methods used. Some of the most popular implementations are GeNeralized Acquisition of Re- current Link (GNARL) [16], Cellular Encoding [17], EPNet [15],Neuro Evolution Augmenting Topologies (NEAT) [18]There are many extensions proposed for NEAT like Real time NEAT (rtNEAT), Hypercube based NEAT (HyperNEAT), cgNEAT (Content-Generating NEAT), Evolutionary Acquisition of Neural Technologies (EANT) [19]. Out of these extensions, HyperNEAT is quite successful since its not task based extension. HyperNEAT is the first extension of NEAT by Stanley and other in 2009 [20]. HyperNEAT is a hypercube based NEAT that can evolves high-dimensional ANNs where the weights of the ANNs are evolved as function of geometry. HyperNEAT is be applied to Robocup [21], multi agent learning [22]. HyperNEAT uses indirect encoding called Compositional Pattern Producing Networks (connective CPPNs).

B. Unconventional Deep Learning Approaches

Traditional ANNs with shallow architectures (ANNs with one hidden layer) have only two levels of computation and learning elements which makes them inefficient to handle training data [23]. Deep architecture is a hierarchical structure of multiple layers with each layer being self-trained to learn from the output of its preceding layer. The learning process of deep architectures termed as 'deep learning' is based on distributed representation learning with multiple levels of representation for various layers. In simple terms, each layer learns a new feature from its preceding layer making the learning process concertized. By this, the learning procedure follows a hierarchy by transforming a low level representation at the first layer to a very high level feature at the last layer with multiple intermediate stages. The knowledge from these intermediate stages can also be utilized as a partial or intermediate solution. Deep architectures empower deep learning strategy using greedy-layer-wise training mechanism which enables to extract only those features that are useful for learning. Apart from layer-wise training, an unsupervised training with unlabeled data makes deep learning successful. Deep architectures requires fewer computational units that allows non-local generalization which result in increased comprehensibility and efficiency that has been proved with its success [24]. According to complexity theory of circuits deep architectures can be exponentially efficient than traditional narrow architectures in terms functional representation for problem solving [23]. Traditionally Artificial Neural Networks (ANNs) are considered as most suitable for implementing deep architectures.

In 1980 Fukushima proposed Neocognition using CNNs [25] which has served as a successful modal for later works on deep architectures. The Fukushima CNNs used unsupervised learning rules to set the initial weights [26] [27] whereas later works improved the concepts by using supervised Back Propagation (BP) for the same [28]. However, Lecun CNN was unable to solve the vanishing or exploding gradients problem that was inherited from ANNs with BP. This problem which is also called as long time lag problem is considered as the fundamental problem of deep learning [29] [30]. In 1998, Lecun used gradient descent training for implementing CNNs which has produced good results in pattern recognition [4] with very time consuming training procedure which is a big concern. Another problem with BP is its mandatory requirement for labelled data at the beginning which is not feasible in case of real world problems.

The Breakthrough in the research of training deep architectures was achieved in 2006 when Lecun, G.E. Hinton and Yoshua Bengio proposed 3 different types of deep architectures with greedy layer-wise training. Lecun used implemented CNNs with greedy layer-wise training and removed the earlier problem of slow learning [31]. Hinton proposed Deep Belief Networks (DBNs) using stacked Restricted Boltzmann machines (RBMs) [32]. A RBM is a stochastic ANN which is capable of learning probability distribution. Yoshus Bengio proposed stacked auto-encoders [33]. A stacked auto-encoder is a multiple layers of auto-encoders stacked in such a way that the output of one layer acts as an input to the other layer. These 3 implementations are considered as the traditional successful approaches for deep learning which formed the base for future implementations.

Generative NeuroEvolution (GNE) for deep learning is developed by implementing HyperNEAT as a feature learner using Compositional Pattern Producing Network (CPPN) for ANNs [34]. CPNN is an encoding procedure of HyperNEAT by encoding weight patterns of ANN to evolve the topology and weights. In traditional HyperNEAT, for given problem, CPPN defines ANN as a solution and its fitness score is determined by evaluating ANNs performance for the problem. Diverging from this traditional approach, GNE trains ANN to learn features by transforming input into features and evaluated by another ML approach by applying to the problems, thus defining the fitness of CPNN. Thus, this process will maximize the performance of the learned solution since HyperNEAT determines the best features out of other ML approach. HyperNEAT without any modifications, generates weight patterns for fully connected feed forward ANNs with only sigmoid activation functions. From this point, the only visible part for HyperNEAT is a geometric coordinate structure of the neurons similar to a graph. ConvNets can be represented in a graph like structure with coordinates of the nodes associated with each other which is similar to HyperNEAT structure. This similarity enables to apply HyperNEAT on ConvNets based architectures. A GA based learning mechanism for DNNs was proposed in 2012, for image classification [35]. In this approach, the layer wise training is performed using GA. Another method, proposed in 2014 is a GA-assisted method for improving the performance of auto-encoder based producing a sparser neural network which produced superior performance than stacked auto-encoders.

In 2013 Yichuan Tang proposed deep learning using Linear Support Vector Machines (LSVMs) [36] which replaces the softmax activation function with LSVM. In ConvNets, softmax is used as an activation function for prediction. A simple ConvNet used with Support Vector Machine (SVM) as the top layer. The training procedure adopted is gradient descent. To improve the generalization capabilities, several models are trained and the results are averaged out. This approach resulted in only a slight improvement. However with LSVMs approach that simply replaces softmax layers with LSVMs has been very efficient and has won ICNL 2013 representation learning challenge for face recognition.

Deep Reinforcement Learning (DRL) uses RL as an initial training mechanism for ConvNets proposed at NIPS2013 [37]. DRL approach is an implementation of deep architecture concept using Reinforcement Learning(RL). The advantages with RL algorithm is its ability to learn from noise and unlabelled data whereas most of the deep architecture implementations requires labelled data [37]. As Most deep learning methods assumes the input data to be independent. RL keeps changing its structures of data distribution with the new learning mechanisms which is not different to deep learning algorithms. This approach proposes a method to overcome the learning challenges of ConvNets for raw video data using RL algorithms. By directly connecting RL algorithms with ConvNets, this approach uses gradient decent updates as training data following learning by experience paradigms. The proposed variant algorithm is tested on Atari 2600 computer games and gave efficient results compared with other approaches.

T-DSN is a deep architecture with stacked multiple blocks on one another with each block is mapped with a bilinear mapping from hidden layers [38]. This mapping is used to incorporate higher order values of the input features. A new training algorithm is proposed with scalable parallel implementation which has outperformed the traditional DNN architecture on TIMIT database [38]. For continues phonetic recognition DNNs need a sequential fine tuning whereas T-DSN produced similar performance without fine tuning.

A combination of CNNs and DBNs called Convolutional Deep Belief Networks (CDBN) achieved significant results on benchmark CIFAR data set. With the objective of optimizing deep learning process especially with deep neural networks, some neural networks techniques like dropout, dropconnect are also used. Other recent developments include Deep Stacking Networks, Spike-and-Slab RBMs, Deep Coding Networks, and Deep Kernel Machines etc.

C. Latest Developments based on feature transfer

A new transductive transference approach was proposed to solve this issue [39]. Transductive learning approaches examines and learns from a specific training to a specific task drawn from the same distribution. So, in case of source and target having different distribution, it follows transductive transference approach, where classification results are improved by transferring exploited labelled training instances from trained network to new network for solving similar problem. Experimental results on using Arabic digits to identity Latin digits (Character recognition) provided improved results both in performance and accuracy. However, the questions about where exactly the generalisation is occurring is still unanswered. Deep Adaptation Network (DAN) architecture, presented in ICML 2015 is the most recent attempt towards understanding the learning process [40] towards generalization of deep CNNs. DAN generalizes the CNN towards domain adaptation scenario where task-specific features are identified and transferred. Further, with DAN it is confirmed that general features can more-over transferable and task specific features are to be tailored to solve a different task.

III. Evolutionary Deep Neural Networks (EDN)

Evolving DNNs is inspired from the NE approach of evolving optimized ANNs. In tradition NE approaches, a population of ANNs are evolved and most fit ANN is selected depending on exit criteria. In some NE approaches multi population based evolutionary strategy is also used. To evolve DNN, we propose to use a novel strategy where two populations with two different co-evolutionary processes. Further, a knowledge extraction process is proposed to extract knowledge from these evolved DNNs. In the case of multi-objective problem solving, it is necessary to produce most optimal ANNs or a family of them to address multiple sub problems which constitute a big problem. In this process maintaining genetic diversity is of paramount importance. One of the major problems of EC is maintaining the diversity of the population. Co-evolution algorithms are not different and are unable to maintain genetic diversity. Numerous methods were proposed to address this problem and some of them include a multi-population approach where new population is generated while evolutionary process.

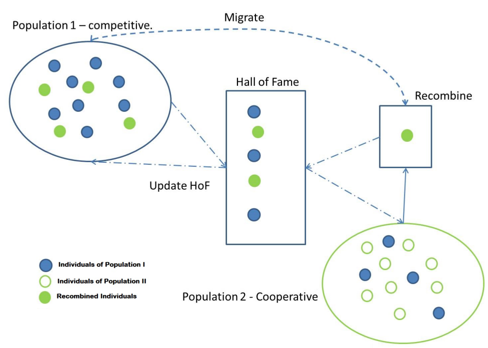

Fig. 1. Evolution Process

There are two types of co-evolution strategies competitive and cooperative. In competitive co-evolution, individuals of a population compete with each other giving rise to an arms race to achieve the top position. With this, it is possible to find a simplest solution among the population of solutions that can win. Cooperative co-evolution evolves solution by recombining individual fragments of sub solutions. Both these approaches have been successful in their respective areas and applications and are equally affected by premature convergence with which genetic diversity will be not maintained after some evolutions. To avoid premature convergence there have been several methods for replacing the individual population, increasing population size, uniform cross over etc. The proposed EDNs with two populations P1 which uses competitive strategy to identify most fit individual and P2 using cooperative strategy to construct optimal solution as shown in Fig.1 . After few generations (or based on some criteria) the migration process involving most fit individual from P1 to P2 or P2 to P1 will be started. This migration process will result in evolving diversified individuals. Using two different types of co-evolutionary processes produces more diversified solutions compared with using same type of co-evolutionary process. Since the individuals of cooperative style of evolution are partial solutions, the mechanism to migrate from cooperative to competitive population with individual as a complete solution, involves an intermediate process called recombine with which partial solutions are recombined to a solution capable of migrating. The problem of individuals re-migrating to their original population is addressed by introducing a unique encoded number with which the origins (population from which it evolved) can be traced back. Further, we propose to implement an automatic self-destruction process at regular intervals to remove non active individuals based on their fitness. We introduce an attribute to encourage combinations with the individuals of other population creating more diversity. A MFI table (Most Fit Individuals) with the list of most fit individuals (from both populations) is maintained similar to hall of fame strategy of NE. The evolution process may be aborted with internal or external exit criteria.

The process of evolving DNNs using NE is practicable and similar to evolving optimized ANNs which has been successful in the past. The evolution of optimized DNNs will address the problem of slow training procedure by providing a warm start to the entire learning procedure. There are still problems concerning a theoretical foundation to DNNs. The aim of this research is not only to develop more efficient evolution of DNNs but also to formulate a theory of DNN as applied to temporal and sequential data classification for future research purposes. In particular, the limits of DNN are not known. One of the tasks will be to evolve DNNs to handle complex, multi-objective optimization involving non-linear relationships between attributes and objective functions.

IV. Initial Experiments

We performed some initial experiments to support our proposed prospect multi-population approach. A 5-layered Deep Neural Network (Nodes: 784,784,784,784,784) is used for the experiment with gradient descent layer wise training followed by a back-propagation for the entire network. The reason for symmetric number of nodes is due to its success in earlier works [41]. Firstly, we carried out the experiments with a 5-layered DNN with random weights (DNN-R). The second strategy is using an evolved DNN with Competitive co-evolution algorithm (DNN-CCEA). For the third strategy we used cooperative co-evolution algorithm to evolve weights (DNN-COCA). We used benchmark MNIST and IRIS data sets for the experiments. Each experiment is carried out for 25 times and the results are presented as Table I and Table II.

Strategy	Accuracy (%)		Error Rate (%)	
	MNIST	IRIS	MNIST	IRIS
DNN-R	94.3	96.9	0.54	0.31
DNN-CCEA	96.3	98.1	0.201	0.19
DNN-COCA	98.7	98.3	0.12	0.131

TABLE I
Experimental results with three strategies

The experimental results classification accuracies and average error of three strategies (DNN-R,DNN-CCEA and DNN-COCA) is presented in Table I. Heuristic random weights approach DNN-R achieved classification accuracy of 94.3% for MNIST and 96.9% for IRIS data sets. Using DNN-CCEA, where weights are evolved using competitive evolution strategy, the classification accuracies are 96.3% and 98.1% whereas for DNN-COCA (co-operative co-evolution) it is 98.7% and 98.3% for MNIST and IRIS data sets respectively. From the results, it is evident that DNN-CCEA and DNN-COCA achieved better classification accuracy which is more than 2% and 4.4% for MNIST and 1.2% and 1.4% for IRIS compared to heuristic random weights approach DNN-R.

Strategy	Avg. Time (hours)	Avg. Performance
DNN-R	14.5	2.5
DNN-CCEA	8.3	1.19
DNN-COCA	10.2	1.73

TABLE II
TRAINING TIME AND PERFORMANCE FOR MNIST

The training time for the experiment is presented in Table II. The training process with DNN-CCEA is faster than DNN-R with a considerable difference of over 6 hours whereas it is about 4 hours for DNN-COCA. It is evident that using evolutionary approaches in tandem with deep learning has speed-up the training process. The main reason for success of this approach is the ability of EC approaches in optimising the weights which provides a warm start to the deep learning process. This approach of using EC techniques to improve the performance of DNNs can be equated to similar strategy used in Neuroevolution approaches. However, the cost of evolving entire DNN compared with the training time of a DNN wit randomly selected topology is to evaluated. But, success of Neuroevolution over randomly selected ANN topology cannot be ignored.

V. CONCLUSION

This paper proposes the prospects of evolving deep architectures which can provide a warm start to the deep learning process. The feasibility of proposed approach is justified by reviewing the some of the implementations of deep architectures using NE and GAs, SVM based learning, Reinforcement Learning. However this work is a direction for reducing learning time for DNNs by using optimized deep neural networks as a starting point which can provide a warm start for deep learning procedure.

Our approaches resulted in an improved classification accuracy of 2% and 4.4%for MNIST data set and 1.2% and 1.4% for IRIS data set compared to heuristic random weights approach. The training process of DNN with EC approach is faster than regular approach with a considerable difference of over 6 hours for MNIST data set. Further, we observed a considerable improvement in the classification accuracies. Our initial experimental results prove that evolutionary approaches provides a warm start to the deep learning, thus, reducing the training time.

Future work includes experimental implementation of EDNs as well as addressing some of the challenges that are identified. One such challenge is deciding the initial topology, weights and biases to start with. As the NE process if more of evolving a network continuously for a particular number of generations, for EDN this may not be feasible since number of generation may not be decided as EDN is problem based approach for evolving optimized ANN. This extends to identifying exit criteria for stopping the process of evolution without which the entire EDN process will be time consuming. The initial experimental implementations and practical implications shows promising results. However, further study is needed to justify the proposed concept of EDNs.

REFERENCES

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, 2006.

[2] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 253–256, May 2010.

[3] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *In NIPS*, 2012.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[5] S. S. Tirumala and A. Narayanan, "Attribute selection and classification of prostate cancer gene expression data using artificial neural networks," in *Inpress*, vol. 9794 of *Lecture Notes in Computer Science*, Springer International Publishing, 2016.

[6] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, pp. 1423–1447, Sep 1999.

[7] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[9] S. Haykin, *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007.

[10] R. H. Crites, A. G. Barto, M. Huhns, and G. Weiss, "Elevator group control using multiple reinforcement learning agents," in *Machine Learning*, pp. 235–262, 1998.

[11] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: a survey of the state of the art," *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*, pp. 1–37, 1992.

[12] Y. Liu and X. Yao, "Evolutionary design of artificial neural networks with different nodes," in *International Conference on Evolutionary Computation*, pp. 670–675, 1996.

[13] X. Yao, "Evolving artificial neural networks," 1999.

[14] P. J. B. Hancock, "Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification," in *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*, pp. 108–122, Jun 1992.

[15] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *Neural Networks, IEEE Transactions on*, vol. 8, pp. 694–713, May 1997.

[16] P. Angeline, G. Saunders, and J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *Neural Networks, IEEE Transactions on*, vol. 5, pp. 54–65, Jan 1994.

[17] F. Gruau, L. C. B. lyon I, O. A. D. D. Doctorat, M. J. Demongeot, E. M. M. Cosnard, M. J. Mazoyer, M. P. Peretto, and M. D. Whitley, "Neural network synthesis using cellular encoding and the genetic algorithm.," 1994.

[18] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, pp. 99–127, June 2002.

[19] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 5, pp. 54–65, 1994.

[20] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artif. Life*, vol. 15, pp. 185–212, Apr. 2009.

[21] D. B. D'Ambrosio, J. Lehman, S. Risi, and K. O. Stanley, "Evolving policy geometry for scalable multiagent learning.," in *AAMAS* (W. van der Hoek, G. A. Kaminka, Y. Lesprance, M. Luck, and S. Sen, eds.), pp. 731–738, IFAAMAS, 2010.

[22] D. B. D'Ambrosio and K. O. Stanley, "Generative encoding for multiagent learning," in *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 819–826, ACM, 2008.

[23] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," in *Large Scale Kernel Machines* (L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, eds.), MIT Press, 2007.

[24] S. S. Tirumala, *Deep Learning: Fundamentals, Methods and Applications*, ch. Unconventional Deep learning approaches, pp. 1–14. USA: Nova Publications, 2016.

[25] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[26] K. Fukushima, "Artificial vision by multi-layered neural networks: Neocognitron and its advances," *Neural Networks*, vol. 37, pp. 103–119, 2013.

[27] K. Fukushima, "Training multi-layered neural network Neocognitron," *Neural Networks*, vol. 40, pp. 18–31, 2013.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems (NIPS 1989)* (D. Touretzky, ed.), vol. 2, (Denver, CO), Morgan Kaufman, 1990.

[29] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München," 1991. Advisor: J. Schmidhuber.

[30] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[31] M. A. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *NIPS*, pp. 1137–1144, 2006.

[32] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, July 2006.

[33] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, U. D. Montral, and M. Qubec, "Greedy layer-wise training of deep networks," in *In NIPS*, MIT Press, 2007.

[34] P. Verbancsics and J. Harguess, "Generative neuroevolution for deep learning," *CoRR*, vol. abs/1312.5355, 2013.

[35] J. Lamos-Sweeney, "Deep learning using genetic algorithms. Master thesis, Institute Thomas Golisano College of Computing and Information Sciences," 2012. Advisor: Gaborski, Roger.

[36] Y. Tang, "Deep learning using support vector machines," *CoRR*, vol. abs/1306.0239, 2013.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[38] B. Hutchinson, L. Deng, and D. Yu, "A deep architecture with bilinear modeling of hidden representations: applications to phonetic recognition," in *ICASSP 2012*, IEEE SPS, March 2012.

[39] C. Kandaswamy, L. M. Silva, L. A. Alexandre, J. M. Santos, and J. M. Sá, *Artificial Neural Networks and Machine Learning – ICANN 2014: 24th International Conference on Artificial Neural Networks, Hamburg, Germany, September 15-19, 2014. Proceedings*, ch. Improving Deep Neural Network Performance by Reusing Features Trained with Transductive Transference, pp. 265–272. Cham: Springer International Publishing, 2014.

[40] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)* (D. Blei and F. Bach, eds.), pp. 97–105, JMLR Workshop and Conference Proceedings, 2015.

[41] S. Tirumala and A. Narayanan, "Hierarchical data classification using deep neural networks," in *Neural Information Processing* (S. Arik, T. Huang, W. K. Lai, and Q. Liu, eds.), vol. 9489 of *Lecture Notes in Computer Science*, pp. 492–500, Springer International Publishing, 2015.