International Journal of Computer Research ISSN: 1535-6698
Volume 23, Number 3

DEEP LEARNING USING UNCONVENTIONAL PARADIGMS

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

ABSTRACT

Deep Learning is gaining more and more popularity due to its success in various applications like Natural Language Processing (NLP), Image recognition and other Machine Learning (ML) paradigms. There are three conventional approaches that formed the basis for deep learning, Convolutional Neural Networks (CNNs), Deep Belief Networks (DBNs) and Stacked Auto-encoders. Following the tradition of other ML paradigms, deep learning also adopted non-conventional approaches for improving the performance. For instance, CNNs and DBNs approaches are combined to form Convolutional Deep Belief Networks (CBDN), which achieved state of art results on CIFAR dataset. Other approaches like deep learning using Linear Support Vector Machines, Deep Q Networks, Deep Kernel Machines, Deep Stacking Networks etc. proved efficient than traditional approaches. Further, improving deep learning process using Evolutionary Computation approaches also achieved limited success. This chapter presents a detailed description of these non-conventional approaches and their applications.

Keywords: Deep reinforcement learning, evolutionary deep neural networks, deep kernel machines, deep q networks, deep svms, genetic algorithms for deep learning

INTRODUCTION

There are three conventional methods for implementing deep architectures. Convolutional Neural Networks (CNNs) proposed by Lecun, Deep Belief Networks (DBNs) by Hinton and stacked auto-encoders by Bengio (Schmidhuber, 2015). To improve the performance of these traditional approaches, new extensions and amalgamations with other Machine Learning (ML) approaches were proposed. For instance, stacked de-noising auto-encoders performed better than traditional stacked auto-encoders on benchmark problems (Vincent, Larochelle, Lajoie, Bengio & Manzagol, 2010). Since DNNs are another form of ANNs, they inherit the drawbacks of ANNs along with their advantages. Evolutionary

* School of Computing and Mathematical Sciences (D-75), Auckland University of Technology, AUT Tower, 2-14 Wakefield Street, Auckland, 1010. E-mail: ssremath@aut.ac.nz

Algorithms (EAs) are used for evolving ANNs to overcome their core problems like overfitting, underfitting etc. and are quite successful with problem solving (Yao, 1999). Though DNNs are quite successful and proven to be better than kernel based methods of DNNs (Bengio & LeCun, 2007), some of the fundamental problems inherited from ANNs still existed. With the objective of optimizing deep learning process, especially with deep neural networks, some neural networks techniques like dropout is proven to be successful (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014). Adding on to this, the demand for continuous improvements in speed and efficiency has provided a new direction for the research.

Combining two or more ML paradigms to achieve improved performance and efficiency is not a new concept. ANNs achieved considerable success when implemented using EAs such as Neuroevolution Augmentation Topology (NEAT), Reinforcement Learning (RL), Quantum computing principles etc. In case of deep learning, these approaches may be categorized into three types, deep learning using (1) conventional deep architecture based methods like CNN, DBN etc. (2) other ML paradigms like RL, SVMs etc. and (3) using Evolutionary Computation (EC) principles.

A combination of CNNs and DBNs called Convolutional Deep Belief Networks (CDBN) achieved significant results on benchmark MNIST dataset (Lee, Grosse, Ranganath & Ng, 2009) and CIFAR dataset (Krizhevsky, Sutskever & Hinton, 2012).

Some of the early developments include Deep Kernel Machines (Cho & Saul, 2009), Deep Coding Network (Lin, Tong, Zhu & Yu, 2010), Spike-and-Slab RBMs, (Luo, Carrier, Courville & Bengio, 2012), Deep Stacking Networks (Hutchinson, Deng & Yu, 2013), Deep Recurrent Neural Networks (Pascanu, Glehre, Cho & Bengio, 2013) etc. Further, machine learning approaches like Reinforcement Learning (Deep Q networks) (Mnih et al., 2015), Deep Support Vector Machines (Tang, 2013) etc. are adopted to improve the performance of deep learning.

On the other hand, EC techniques are quite adoptive to many machine learning approaches and deep learning is not an exception. EAs are used to optimize deep neural networks (similar to neural networks). Implementation of genetic algorithms and NEAT based weight optimizations are successful in enhancing the efficiency of deep learning. The success of implementing evolutionary algorithms for DNNs for performance enhancement is widely acknowledged (Tirumala, 2014). Using Genetic Algorithms (GAs) for deep learning was first proposed in 2012 (Lamos-Sweeney, 2012). Using evolutionary algorithms like NEAT (Verbancsics & Harguess, 2013) to evolve DNNs, achieved limited success. Using GAs for evolving DNNs is not challenged against any benchmark problems and especially it has not been compared against the state-of-art approaches. However, it is noteworthy to understand that implementing deep learning on optimized (evolved) DNNs will provide a warm start to the entire process of deep learning.

When aiming at improving the speed of deep learning process, the importance of application domain should not be neglected. Since EAs are efficient in solving optimization problems, it is evident that DNNs with evolutionary strategies will follow the same standards of EA. Currently the application of deep learning is limited to natural language processing, image recognition and other related aspects. However, research trends indicate that, the power of DNNs is not fully realised yet. Currently, application of DNNs for opinion mining (Irsoy & Cardie, 2014), taxonomic data classification (Tirumala & Narayanan, 2015) and sentimental

analysis for online short messages like Twitter, Facebook, etc. (dos Santos & Gatti, 2014) is gaining momentum.

DEEP LEARNING APPROACHES

Traditional Deep Architecture-Based Approaches

Convolutional Deep Belief Networks (CDBN)

Convolutional Deep Belief networks are constructed using Convolutional Restricted Boltzmann machine (CRBM) similar to DBN, which uses RBMS. Unlike in RBM, the weights are shared between the layers, which makes CRBM, a standout model. A CRBM is similar to RBM with two layers and input layer V) with NvXNv binary units and a hidden layer NH2 xK hidden binary units where K is the number of groups with H with NHXNH = NH2 units in each group (Lee et al., 2009). The main advantage of CDBN lies in its implementation of probabilistic max-pooling. Conventional CNNs use shrinking mechanism on the representations in pooling layers. The use of max-pooling allows to shrink the representation, high level representations are least affected by variance, reducing the computational burden which makes CDBN more efficient (Lee et al., 2009). CDBN proved to be better than conventional DBNs with both labelled and unlabelled data (Lee et al., 2009) and with complex dataset like CIFAR (Krizhevsky et al., 2012).

Deep Kernel Machine (DKM)

Stacking various kernels similar to layers of neural networks constitute DKMs. The arrangement can be either linear or non-linear. Extending principle component analysis (PCA) to work using kernel methods is called kernel PCA. Standard PCA linear operations are not performed using kernel methods in reproducing Hilbert space.

In DKM, the learning (deep learning) is done by applying kernel PCA method iteratively on each layer. For each layer, the input to the Kernel PCA results in an output (features). This output serves as input to the kernel PCA of the next layer, similar to regular RBM based deep learning approach (Cho & Saul, 2009). For each layer in DKM, a supervised feature selection is performed followed by unsupervised kernel PCA. This feature selection is useful to eliminate unwanted noisy features. The classification in DKM is performed in the final layer using kNN.

The main aim of DKM is to utilize kernel-based approach for multiclass classification problems. The experiment involves two benchmark multiclass datasets. DKM outperformed previous SVM based methods in classifying these datasets. This approach provided a new direction on adopting deep learning using kernel based approach for DKM. Though DKM's performance is considerable against SVMs, this approach could not overthrow the efficiency of traditional DBMs. However, DKM provides a direction for implementing deep learning using kernel method.

Deep Coding Network

Deep coding network is an extension of sparse coding network by applying multiple - layer (Lin et al., 2010). Deep coding network uses DBN based architecture with hierarchical

structure. Traditional single layer sparse coding network provides raw approximation of data. By extending it with multi-layer hierarchical structure, the extended layer can provide in-depth local search for each function avoiding the possibility of overfitting. Since the search is looking at the higher layers with non-zero coefficients of the previous layer, this is faster and efficient compared with flat coding. This is due to the nature of hierarchical structure, which in turn reduces the computational cost compared to regular sparse coding network. The proposed approach uses a fitting mechanism with multiple small models for each layer in a multi-layer environment, instead of an overall single layer approach. As the fitting is applied on multiple layers, each process can work in parallel, which increases its speed furthermore and will be more effective for big datasets. When deep coding network is applied for MNIST dataset, the classification error is reduced and performance has increased significantly compared with traditional single layer approach. The first layer's context of information is mapped with the second layer's depth details resulting in more classification accuracy. Similar results were achieved when 2-layer deep coding network is applied to PASCAL-2007 dataset. However, the performance and efficiency of a network with more than two layers is not addressed in this research.

Deep Stacking Networks

Deviating from the traditional approach of linear weight update mechanism, Deep Stacking Network (DSN) uses parallel weight learning procedure, in which, a block-wise supervised training is adopted instead of overall BP training, a regular deep learning training mechanism (Deng & Yu, 2011). A DSN consists of a single hidden layer. The input layer and hidden layer are linked by a lower weight matrix and the logistic sigmoid hidden layer (non-linear) is connected to the output layer (linear) with an upper-layer weight matrix (Hutchinson et al., 2013). DSN weights are iteratively updated using gradient-descent. Further, the output of each DSN is expressed as output label vector, which is then concatenated with every block input to the next DSN, which results in improved performance for each layer (DSN block). Each layer of DBN tries to identify and resolve the transformed features where as in DSN, this particular step is eliminated, resulting in more optimized weights.

T-DSN (Tensor-DSN), an extension to DSN is proposed with 3 fixed blocks of DSNs (Hutchinson et al., 2013). In DSN, there is only one set of lower level weight matrix (input-hidden) whereas T-DSN uses two sets of lower level weight matrices to connect to the hidden layer (in parallel). This makes T-DSN faster and more efficient. Furthermore, T-DSN has an advantage of parallel implementation with multiple CPUs.

Spike-and-Slab RBM (ssRBM)

Spike and Slab RBM is formed by introducing a real valued vector (slab) and a binary valued variable (spike) for each unit in the hidden layer (Courville, Bergstra & Bengio, 2011). This new way of representation enables to select data representation from binary and continuous. This choice helps to decide the representation based on task, providing an advantage over regular data representation for image classification tasks, when the property representing the intensity of the input values is shared with rectified linear units. This further standardizes the image intensity variable making it more accurate. The ssRBM uses similar learning style of RBM but differs in the selection of samples for training.

The ssRBM uses a rejection scheme to remove unwanted samples. During training, the instability in the sample selection is avoided by initializing the parameters within stable

regime of space. Further, decreasing learning rate using ssRBM training increase the stability of the parameter space. All these strategies in the training ensures stability and efficient sampling.

A deep network in convolutional-based architecture is built using ssRBM for pattern recognition in CIFAR-10 dataset. Experimental analysis is conducted to understand the effect of spike and slab variables. By comparing the previous works on CIFAR dataset, ssRBM produced better and efficient results compared with standard RBMs as well as modified RBMs.

Machine-Learning-Based Deep Learning Approaches

Deep Q-Networks (DQN)

Applying deep learning requires large training data whereas RL works on reward-based algorithms with noisy data. Moreover, RL differs with supervised learning in terms of time delay between action and rewards. Deep learning assumes a fixed feature distribution while RL uses the learn-as-you-go method. Further, in deep learning, the training is performed on samples assumed to be non-related while it is the opposite for RL. On the other hand, Q-learning used to select the most possible action for a finite process, which standard RL lacks. When Q-Learning is used for selecting and updating weights of a standard DNN, it is called Deep Q Network (DQN) (Mnih et al., 2013). This new learning procedure, has an advantage of higher efficiency. Since the deep learning procedure relies on sequential sample selection, a randomised selection using Q-learning reduces frequency and variance of weights updates. The samples for next training are determined by parameters in the current layer (similar to selection of reward for action). In other words, Deep Q-learning with an amalgamation of DNNs and Q-learning is expected to be more efficient than regular RL paradigms.

Initially this approach is implemented for Atari gaming and the result shows that deep architecture based RL DQN, performs better than all other existing RL based approaches (Mnih et al., 2013). Recent implementation of DQN achieved near human level control for Atari game playing with the learning agent producing optimal game strategy for most of the games (Mnih et al., 2015). This experiment further proved that DQN is capable of producing long-term plans and strategies for certain games like Breakout. In this bricks breaking game, the agent using DQN was able to produce a strategy to move around and break more bricks instead of hitting nearest bricks that yield less points (Mnih et al., 2015). So, it is very important to understand the advantage of using DQN instead of RL, and at the same time, it is important to note that DQN implementation is highly dependable on reply algorithm that acts as the answer for an action. So, it is evident that DQL, a deep architecture based RL is advantageous than standard RL.

Deep Support Vector Machines

There are two types of SVM implementations for deep learning. In the first method, a deep architecture is constructed using SVMs as the replacement for ANNs, which is termed as Deep SVMs (Wiering, Schutten, Millea, Meijster & Schomaker, 2013). In the second method, SVMs are used as an alternative for Softmax function used in DNNs (Tang, 2013).

Deep SVM is constructed by linearly arranging SVMs, first set for learning features from the input vectors and the second SVM called main SVM for prediction (Wiering et al., 2013).

This architecture is supposed to be superior to regular SVM, since regular SVM has single adjustable layer which limits its performance. DVSM is trained with gradient ascent and descent. The learner SVMs learn to minimize the objective whereas the main SVM tries to maximize it. The regularization capability of main SVM avoids overfitting and the learner SVMs optimize the performance. Further, to make the learning more efficient, gradient ascent algorithms adopt penalties.

In the second approach, a convolution based approach is used with one vs all approach at the top using linear SVMs. This approach is similar to CNNs in terms of architecture. However, in-line with CNNs, it is necessary to construct and train different models to obtain a fair generalization. For training, stochastic gradient descent was used. This model is tested against Softmax based CNN model for image recognition. The images are preprocessed by subtracting individual mean value before normalising it to 100. The value of each pixel is adjusted to the standard deviation value across the entire dataset. This model won the ICML workshop - 2013 with a validation score of 69.4%. Further, this model achieved better performance on MNIST dataset as well compared to its Softmax counterpart.

Evolutionary Computation-Based Deep Learning

Using nature inspired computation for improving the performance and efficiency of ML algorithms is not new. Especially with ANNs, genetic and evolutionary based strategies achieved considerable success (Xu, Moriguch & Honiden, 2013; Hausknecht, Lehman, Miikkulainen & Stone, 2014). However, implementing EC principles for DNNs has achieved limited success in the verse of development. Optimization of ANNs is achieved by evolving topology or weights or both. This is not different in the case of DNNs and recent works utilized these strategies. Two of the three approaches discussed below use genetic algorithm based work whereas the third approach uses HyperNEAT, an extension of NEAT.

Deep Learning Using Genetic Algorithm

In the attempt to use GA for deep learning, initially a near zero valued matrix is used for training, keeping identity matrix as the target. This sequence of values is used as the initial input gene with a magnitude parameter (Lamos-Sweeney, 2012). RBM based network is used for this approach, replacing the gradient descent training with GA. The main principle behind this implementation is the mechanism of error reconstruction using encoding and decoding. Mean square error (MSE) between input and the reconstruction is an obvious choice for the algorithm to calculate fitness function.

Initially, a population of genes (matrices) is generated with randomly selected gene. Cross over is performed by randomly selecting two elements within a matrix after linearizing the matrix. Mutation is performed by replacing randomly selected element with a new element by comparing the fitness. To control the size of the population, a penalty function is introduced which stops generating ineffective genotypes. This sparse network based operation is tested with image data including handwritten data, face recognition and cat image data and has achieved considerably good results. Further, by observing the results, the reconstruction mechanism is proved to be working (Verbancsics & Harguess, 2013).

However, the main issue with this algorithm is, its limitations being not mentioned. For instance, when this algorithm is tested against a small dataset, the theoretical limitations are

not mentioned. And moreover the results are not compared with state-of-art results of other deep learning algorithms on benchmark datasets. It made this attempt a good starting point to justify the possibility of using GA for training, but not as a standardized process that can be compared with other successful deep learning approaches.

Genetic Algorithm-Assisted Deep Learning

GA assisted approach for deep learning is, modifying auto-encoder based deep learning to improve its performance. In traditional auto-encoder based method, the decoding layer weights are knotted against the encoder layer weights. These weights are in a fixed set. In this approach, a single set of weights is treated as chromosome and GA method is used to generate multiple sets of weights as a population for the auto-encoder. For each chromosome, RMSE is calculated for the training samples, with 1/RMSE as the fitness value of individual chromosome. Chromosomes with least fitness values are removed from the population, and fit chromosomes are used as weights for the auto-encoder. New chromosomes are evolved using mutations among fit individuals selected randomly to replace the removed individuals in the population. BP is used to train the network and update the weights. Once auto-encoder is trained, from the surviving set of chromosomes, most fit one serves as the weights for that layer. This process is continued for each layer as every individual layer is trained separately.

The experiment is performed using a 5-layer auto-encoder network with 10-chromosome population on MNIST dataset. When results are compared with BP based auto-encoder network, the GA assisted method produced least reconstruction error. To compare the quality using supervised SVM classification, GA assisted method achieved superior performance with less classification error compared to traditional auto-encoder approach with BP.

However, the difference between the classification errors is 0.04%, which shows that the GA assisted method has very minimal advantage compared with traditional methods on classification accuracy performed on 10,000 samples (around 17% of MNIST dataset) which make these results questionable. Further, GA assisted method is far behind the state-of-art results on MNIST dataset using CNNs. Moreover, the evolution process used does not guarantee superior weights or performance when compared to state-of-art results in deep learning.

Neuroevolution for Deep Learning

HyperNEAT, an extension to the NEAT algorithm can be used for weight patterns generation for feed-forward ANN with sigmoid as its activation function. A graph based geometric coordinate system is consistent visible part for HyperNEAT. In other words, the ANN layers form one axis, weights as the second axis and features as the third axis in the coordinate system architecture. This is similar to CNN structure. The trained procedure in HyperNEAT is similar to any other feed-forward ANN.

A new attempt to implement deep learning using HyperNEAT on image data and layer-wise training (deep learning), a hypercube with 8 dimensions is used with features as f-axis, neuron constellation (layer) with individual features as x-axis and y-axis representing individual pixel location (similar to CNNs for image classification by Lecun.) The topology used is a multilayer ANN travelling along z-axis with Compositional Pattern Producing Networks (CPPN) representing the connections in the four dimensional substrate. Four co-

ordinates x; y; f; z are used to determine the position of each neuron. This algorithm is experimented with CNN topology, LeNet-5 used by Lecun at his first attempt on deep leaning (Lecun, Bottou, Bengio & Haffner, 1998). When tested on MNIST dataset, CNN based HyperNEAT performed better than the original HyperNEAT.

This procedure, which uses HyperNEAT based training instead of deep learning approach, uses LeNet-5 architecture instead of conventional ANNs. Therefore, this serves as a comparison between standard HyperNEAT and HyperNEAT on LeNet5. This cannot be considered with pure CNNs on MNIST dataset that still holds the state-of-art results. However, this can be considered as a good attempt to implement EC based algorithm on DNNs.

WHAT'S NEXT

The main purpose of combining two or more methods may not be always successful. The sole purpose of these amalgamations is to improve the performance and efficiency, at-least in current context. Present developments in Artificial Intelligence (AI) draw their inspiration from their theoretical foundations of 1950's. After the development of powerful computing machines, it was possible to test and try different AI and ML approaches. For instance, the potential of ANNs was not completely revealed until recent times. It is the same with quantum computing. Though DNNs proved to be producing state-of-art results in most of the ML challenges, there are still many open questions about utilizing the full capability of deep learning. The availability of large volumes of data, demands the emergence of powerful ML techniques for data extraction and knowledge discovery. Big data analysis requires huge processing in terms of hardware, while algorithms pose unprecedented challenges to ML algorithms. Deep learning provides, answers to a certain extent to the challenges posed by Big data. The possibility of implementing deep learning algorithms for Big data is undergoing extensive research (wen Chen & Lin, 2014).

The concept of real time data analysis, especially with temporal Big data is another challenge which needs to be addressed. Considering the current technological advances, deep learning might be the ideal choice for accurate analysis, which has already proved its potentiality against various ML challenges like natural language processing, image recognition, face recognition, text mining and sentiment analysis.

Quantum computing is by far considered to be superior to any other computing systems. Quantum inspired computing has been quite successful for ANNs and EC. It will be interesting to see how a complex quantum computing approach can be used for deep learning, which is a complex learning process. Another interesting aspect of deep learning is implementing quantum principles in deep learning, especially for feature extraction. Considering the success of quantum computing and deep learning, this combination may serve as the new horizon in the prospect of Machine Learning.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

REFERENCES

Bengio, Y. & LeCun, Y. (2007). Scaling learning algorithms towards ai. *Largescale kernel machines*, *34*(5).

Cho, Y. & Saul, L. K. (2009). Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams & A. Culotta (Eds.), *Advances in neural information processing systems 22* (pp. 342–350). Curran Associates, Inc.

Courville, A. C., Bergstra, J. & Bengio, Y. (2011). A spike and slab restricted boltzmann machine. In G. J. Gordon, D. B. Dunson & M. Dudk (Eds.), *Aistats* (Vol. 15, p. 233-241). JMLR.org.

Deng, L. & Yu, D. (2011, August). Deep convex network: A scalable architecture for speech pattern classification. In *Interspeech.* International Speech Communication Association. Retrieved from http://research. microsoft.com/apps/pubs/default.aspx?id=152133.

Dos Santos, C. N. & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th international conference on computational linguistics (coling), dublin, ireland.*

Hausknecht, M., Lehman, J., Miikkulainen, R. & Stone, P. (2014). A neuroevolution approach to general atari game playing. *Computational Intelligence and AI in Games, IEEE Transactions on*, *PP*(99), 1-1. doi: 10.1109/TCIAIG.2013.2294713.

Hutchinson, B., Deng, L. & Yu, D. (2013, August). Tensor deep stacking networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, *35*(8), 1944–1957. Retrieved from http://dx.doi.org/ 10.1109/TPAMI.2012.268 doi: 10.1109/ TPAMI.2012.268.

Irsoy, O. & Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 720–728). Retrieved from http://aclweb.org/anthology/D14-1080.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems.*

Lamos-Sweeney, J. (2012). *Deep learning using genetic algorithms. master thesis, institute thomas golisano college of computing and information sciences.* (Advisor: Gaborski, Roger).

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the ieee* (pp. 2278– 2324).

Lee, H., Grosse, R., Ranganath, R. & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning* (pp. 609–616). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1553374.1553453 doi: 10. 1145/1553374.1553453.

Lin, Y., Tong, Z., Zhu, S. & Yu, K. (2010). Deep coding network. In *Advances in neural information processing systems* (pp. 1405–1413).

Luo, H., Carrier, P. L., Courville, A. C. & Bengio, Y. (2012). Texture modeling with convolutional spike-and-slab rbms and deep extensions. *CoRR, abs/1211.5687*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Hassabis, D. (2015, February 25). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. doi: 10.1038/nature14236.

Pascanu, R., Glehre, a., Cho, K. & Bengio, Y. (2013). How to construct deep recurrent neural networks. *CoRR, abs/1312.6026*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85-117. (Published online 2014; TR arXiv:1404.7828) Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929-1958. Retrieved from http://jmlr.org/papers/v15/srivastava14a.html.

Tang, Y. (2013). Deep learning using support vector machines. *CoRR, abs/1306.0239*.

Tirumala, S. S. (2014). Implementation of evolutionary algorithms for deep architectures. In *Proceedings of the 2nd international workshop on artificial intelligence and cognition (AIC), torino, italy, november* (pp. 164–171). Retrieved from http://ceur-ws.org/Vol-1315/paper15.pdf.

Tirumala, S. S. & Narayanan, A. (2015, Nov). *Hierarchical data classification using deep neural networks.* (Accepted for 22nd International Conference on Neural Information Processing (ICONIP2015)).

Verbancsics, P. & Harguess, J. (2013). Generative neuroevolution for deep learning. *CoRR, abs/1312.5355*.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.A. (2010, December). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, *11*, 3371–3408. Retrieved from http://dl.acm.org/ citation.cfm?id=1756006.1953039.

Wen Chen, X. & Lin, X. (2014). Big data deep learning: Challenges and perspectives. *Access, IEEE*, *2*, 514-525. doi: 10.1109/ACCESS. 2014. 2325029.

Wiering, M., Schutten, M., Millea, A., Meijster, A. & Schomaker, L. (2013). Deep support vector machines for regression problems. In *Proceedings of the international workshop on advances in regularization, optimization, kernel methods, and support vector machines: theory and applications.*

Xu, S., Moriguch, H. & Honiden, S. (2013, June). Sample efficiency analysis of neuroevolution algorithms on a quadruped robot. In *Evolutionary computation (cec), 2013 ieee congress on* (p. 2170-2177). doi: 10.1109/CEC.2013.6557826.

Yao, X. (1999, Sep). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*(9), 1423-1447. doi: 10.1109/5.784219.