Multi-Layer Neural Networks for Quality of Service oriented Server-State Classification in Cloud Servers

Yonghua Yin, Lan Wang and Erol Gelenbe *FIEEE*
Intelligent Systems and Networks Group
Electrical & Electronic Engineering Department
Imperial College, London SW7 2AZ, UK

Abstract—**Task allocation systems in the Cloud have been recently proposed so that their performance is optimised in real-time based on reinforcement learning with spiking Random Neural Networks (RNN). In this paper, rather than reinforcement learning, we suggest the use of multi-layer neural network architectures to infer the state of servers in a dynamic networked Cloud environment, and propose to select the most adequate server based on the task that optimises Quality of Service. First, a procedure is presented to construct datasets for state classification by collecting time-varying data from Cloud servers that have different resource configurations, so that the identification of server states is carried out with supervised classification. We test four distinct multi-layer neural network architectures to this effect: multi-layer dense clusters of RNNs (MLRNN), the hierarchical extreme learning machine (H-ELM), the multi-layer perceptron, and convolutional neural networks. Our experimental results indicate that server-state identification can be carried out efficiently and with the best accuracy using the MLRNN and H-ELM.**

I. Introduction

The performance of a set of specific servers in large computing infrastructures such as the Cloud [1], [2], can be difficult to predict, since it is affected by the servers' configuration (number and speed of the processors, memory size, etc.), as well as by workload characteristics (such as compute or I/O bound, execution times, IO transfer sizes and times), and the dynamic and statistical behaviour of the set of tasks being run, including variations in memory usage, disk usage, etc., caused by random numbers of users with varying demands on computing resources. On the other hand, end users expect that Cloud services will respect Service Level Agreements (SLA) so that an accurate prediction of server performance and Quality of Service (QoS) is of great importance.

Thus, machine learning techniques have been used in several contexts [3], and the work in [4] has conducted feature selection from data collected in servers and then applied support vector machines (SVM) to detecting anomalies.

It was also shown [2] that machine learning can be used to predict the occurrence of unexpected events based on system-feature measurements.

On the other hand, previous work [5]–[7] exploited reinforcement learning with the Random Neural Network (RNN) [8], [9] for dynamic workload allocation in the Cloud and in networked systems. Initially developed to represent biological neurons [10], the RNN has also been used for handling various tasks through its learning capability [11]–[18].

While conventional deep-learning tools such as the multi-layer perceptron (MLP) and the convolutional neural network (CNN) have achieved great successes in machine learning in recent years [19]–[22], the extreme learning machine (ELM) which is a one-hidden-layer neural network has also been generalized into multi-layer architectures, and hierarchical extreme learning machine (H-ELM) has been shown to be an efficient tool [23], [24].

Recent work [25], [26] has connected the RNN to deep learning [19]–[22] and to the ELM concept [23], [24]. Also, a new RNN training procedure with a multi-layer architecture of dense clusters of RNN (MLRNN) with soma-to-soma interactions was proposed [25]–[27], providing fast and very good generalisation performance [27].

In this paper, we investigate the feasibility of these tools to infer the states of servers in Cloud system. Specifically, we investigate the MLRNN, H-ELM, MLP and CNN.

First, we present a procedure to construct classification datasets by collecting time-varying system information in servers with different configurations in the Cloud. The collected datasets are labelled with the execution time of a baseline task in a given server to represents the different server states. These datasets are then used to transform the problem of detecting the servers' states in the Cloud into a supervised classification problem which neural networks are good at handling.

Then, we apply the above mentioned four types of multi-layer neural networks to solve these classification problems. Numerical results seem to indicate that these specific classification problems related to server-state detection can be solved with a high degree of accuracy using all of these neural-network tools, but that the MLRNN and H-ELM show more promise because they result in significantly greater efficiency.

II. Classification dataset construction

A dataset for supervised learning is generally composed of two parts, the input-attribute matrix X and the label Y that can be a vector or a matrix depending on the types of the learning problem. In this section, we construct X by

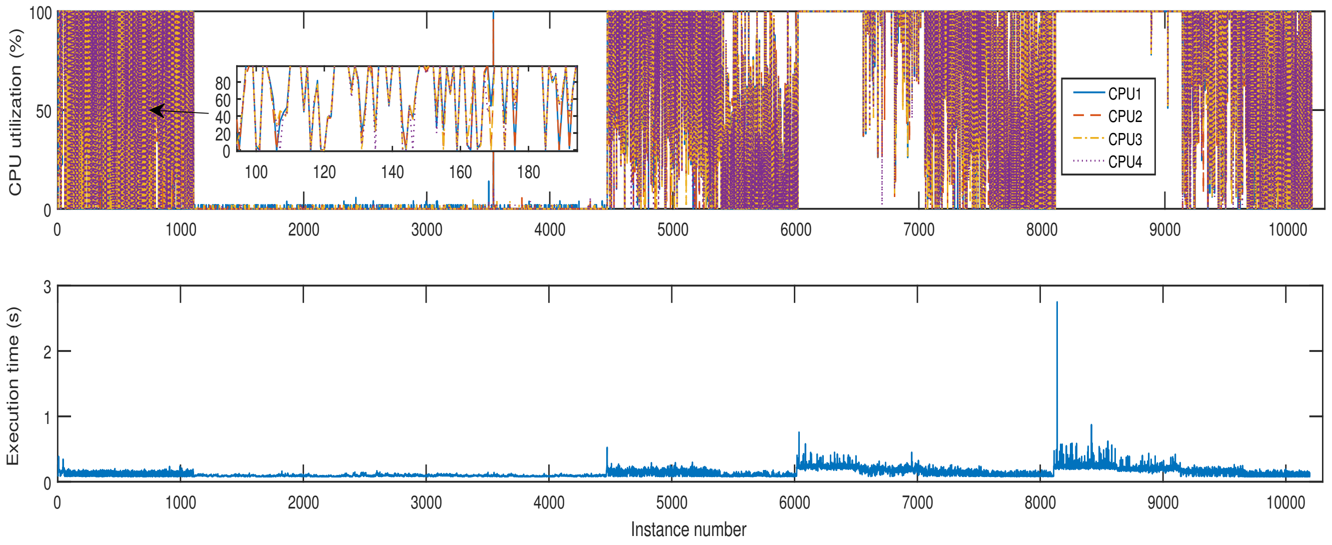

Fig. 1. CPU utilization percentage and execution time collected in Server 1 using Task 1 under stochastic background loads.

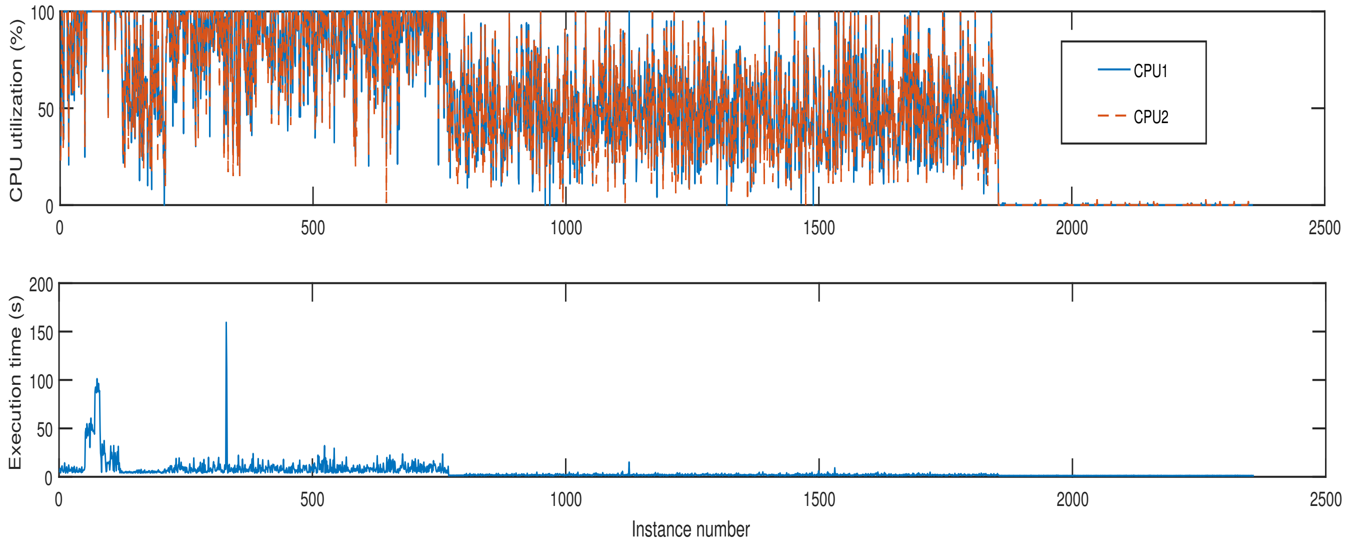

Fig. 2. CPU utilization percentage and execution time collected in Server 2 using Task 2 under stochastic background loads.

collecting the time-varying system information in a server, while, for the label information Y, we introduce baseline tasks and measure the execution time required for the server to finish the task.

A. Input-attribute data collection

Time-varying information from the server concerning its CPU, memory, swap memory and disk usage are collected, with the number of dimension denoted as n that can be different for different servers. During the collection, we can periodically change the background load via simulating dynamic stochastic occupation of computing resources by stochastic number of users.

B. Label information collection

We conduct baseline tasks on the server with the execution time recorded, where the baseline task includes operations of matrix multiplication and pseudoinverse and is used to test the server state (i.e., how fast the server can handle the task). Two baseline tasks are used for label information collection, where Task 1 is simpler that requires less computational time than Task 2.

The execution time (s) required to finish a baseline task is denoted by t_{exe}. Rather than solving a regression problem with continuous execution-time labels, we assign discontinuous class labels y to the instances according to t_{exe} using a pre-setting interval vector α and therefore transform the problem into a classification problem, e.g., $\alpha = [a, b, c]$

TABLE I
ATTRIBUTE, INSTANCE AND CLASS NUMBERS OF CLASSIFICATION DATASETS COLLECTED FROM DIFFERENT SERVERS

Dataset	Attribute No	Instance No.	Class No.
Server1-T1	72	10196	5
Server2-T2	63	2358	3
Server3-T2	63	927	3

means: $y = 1$ if $0 \leq t_{\text{exe}} < a$; $y = 2$ if $a \leq t_{\text{exe}} < b$; $y = 3$ if $b \leq t_{\text{exe}} < c$; $y = 4$ if $t_{\text{exe}} \geq c$.

C. Dataset construction and visualization

After the data collection, a classification dataset is constructed in the following manner. Three collections of the time-varying information right before conducting a baseline task is used as input attributes for an instance. Then, the dimension of input attributes is $3n$. The labels are the values of y described in Subsection II-B.

Three servers with different configurations are created in Google Cloud Platform: Server 1 has 4 CPUs and 3.6 GB memory; Server 2 has 2 CPUs and 7.5 GB memory; Server 3 has 1 CPU and 0.6 GB memory. For better illustration, we visualize several attributes in the datasets (i.e., CPU utilization percentages) and the corresponding labelling information (i.e., the execution time). Figure 1 visualizes the dataset collected in Server 1 using Task 1, while Figures 2 and 3 are for Server 2 using Task 2 and Server 3 using Task 2, respectively. From these figures, we can see the dynamic changes of background loads in the servers. In addition, the attribute, instance and class numbers of these classification datasets are listed in Table I.

III. CLASSIFICATION ACCURACIES COMPARISON

In this section, we compare classification performances of four types of multi-layer neural networks to handle the classification problems related to the classification datasets collected in servers illustrates in Subsection II-C. Each classification dataset is randomly and equally separated into a training dataset and a testing dataset. Note that we do not conduct any preprocessing to the input attributes of the dataset.

Let us first consider the Server1-T1 dataset for Server 1 shown in Table I. In these numerical experiments, we use the MLRNN [25]–[27], and the MLP [28] with the dropout technique [29], the CNN also with dropout [28], [29] and H-ELM [24]. The structures of the MLRNN, MLP and H-ELM are 72-500-1000-5. For the CNN, there are two convolution layer with 10 convolution filters, a pooling layer, a fully connected layer and an output layer. For the MLP and CNN, we use the ReLU [30] and Tanh activation functions. The corresponding classification results are given in Table II. The results of the MLRNN and H-ELM are obtained by conducing 100 trials, where the testing accuracies of all 100 trials are given in Figure 4 for further comparison. As seen from the table, the MLP and CNN with ReLU activation barely converge and using Tanh activation is much better than using ReLU activation in this case, where the reason may be that the nonlinearity of ReLU is not sufficient for learning the complex mapping of the Server1-T1 dataset while Tanh is. The MLRNN achieves the highest testing accuracy for state detection of Server 1 among the compared deep-learning tools. Moreover, the training time of the MLRNN and H-ELM is significantly less than that of the conventional MLP and CNN. From Figure 4, we can see that, most trials of the MLRNN achieves higher testing accuracies than the highest one of the H-ELM among all trials.

TABLE II
TESTING ACCURACIES (%) AND TRAINING TIME (S) OF DIFFERENT METHODS FOR STATE DETECTION OF SERVER 1 WITH TASK 1.

Method	Testing accuracy	Training time
MLP (ReLU) [28]	6.79	938.14
MLP (Tanh) [28]	73.64	527.88
CNN (ReLU) [28]	1.18	220.28
CNN (Tanh) [28]	77.17	586.36
H-ELM [24]	79.80	0.28
MLRNN	**80.35**	1.59

TABLE III
TESTING ACCURACIES (%) AND TRAINING TIME (S) OF DIFFERENT METHODS FOR STATE DETECTION OF SERVER 2 WITH TASK 2.

Method	Testing accuracy	Training time
MLP (ReLU) [28]	33.33	229.17
MLP (Tanh) [28]	81.09	129.93
CNN (ReLU) [28]	33.16	39.65
CNN (Tanh) [28]	81.51	142.78
H-ELM [24]	**87.36**	0.05
MLRNN	86.60	0.31

Then, we consider the Server2-T2 and Server3-T2 datasets for Servers 2 and 3 shown in Table I. In these experiments, we also use the same neural-network tools with slightly different structures. For the MLRNN and HELM, the structures are 63-50-10-1000-3. For the MLP and CNN, the same structures as those for the Server1-T1 dataset are used here. The results are given in Tables III and IV and Figures 5 and 6. We can see that for these two datasets, the H-ELM is the most efficient among the four tools.

These results verify the feasibility of the multi-layer neural networks for server-state detection in the Cloud, and the MLRNN and H-ELM achieve higher accuracies and are more efficient than the MLP and CNN in handling the related tasks.

IV. CONCLUSION

In this paper, a procedure has been presented to construct classification datasets related to state detection of servers in the Cloud. The purpose is to transform the inference problem into a supervised classification problem.

Four different types of deep neural networks have been applied to solving these classification problems, including the

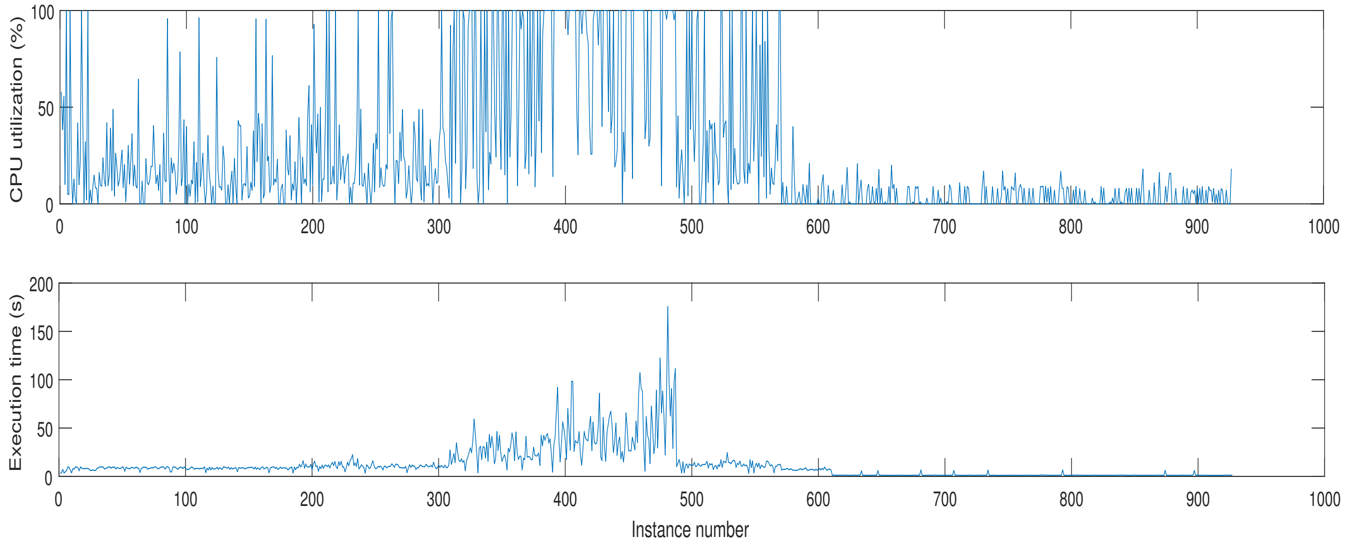

Fig. 3. CPU utilization percentage and execution time collected in Server 3 using Task 2 under stochastic background loads.

TABLE IV
TESTING ACCURACIES (%) AND TRAINING TIME (S) OF DIFFERENT METHODS FOR STATE DETECTION OF SERVER 3 WITH TASK 2.

Method	Testing accuracy	Training time
MLP (ReLU) [28]	33.05	103.65
MLP (Tanh) [28]	69.76	66.98
CNN (ReLU) [28]	32.18	16.79
CNN (Tanh) [28]	85.10	47.81
H-ELM [24]	**88.55**	0.03
MLRNN	83.80	0.11

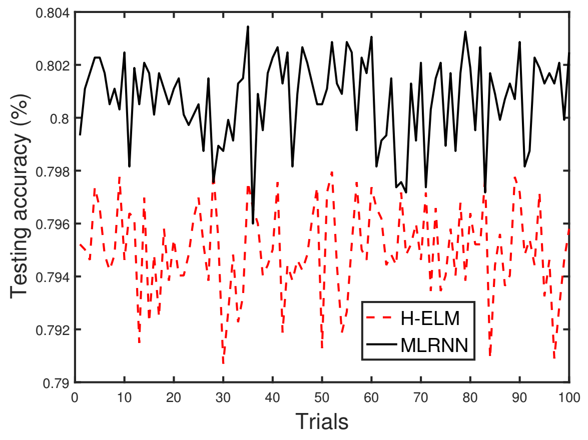

Fig. 4. Testing accuracies of all 100 trials of H-ELM and MLRNN for state detection of Server 1 with Task 1.

MLRNN that is recently proposed based on the ideas of dense clusters, the H-ELM and the conventional deep-learning tools of MLP and CNN.

These neural-network tools have then been compared by using experimental data from three different servers.

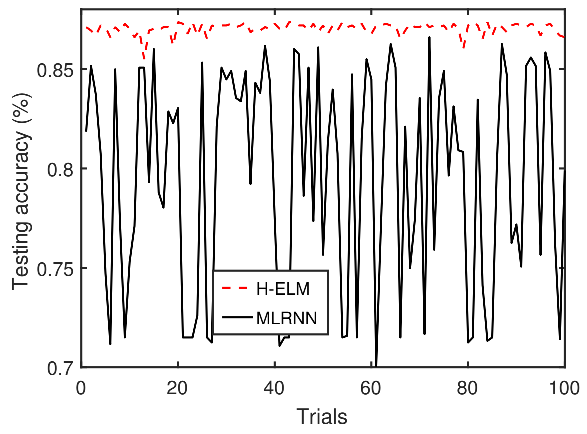

Fig. 5. Testing accuracies of all 100 trials of H-ELM and MLRNN for state detection of Server 2 with Task 2.

The results demonstrate that the classification problems related to server-state detection can be solved in high accuracies using these neural-network tools and the MLRNN and H-ELM show very high efficiency compared with the other two deep-learning tools.

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the EC 7th Framework Program PANACEA Project, Grant Agreement No. 610764, to Imperial College London.

REFERENCES

[1] J. Doncel, U. Ayesta, O. Brun, and B. Prabhu, "A resource-sharing game with relative priorities," in *IFIP Performance 2014, Turin, Italy (2014)*, 2014.
[2] P. D. Sanzo, A. Pellegrini, and D. R. Avresky, "Machine learning for achieving self-* properties and seamless execution of applications in the cloud," in *NCCA 2015, Munich, Germany*. IEEE, 2015.
[3] L. Wang and E. Gelenbe, "Experiments with smart workload allocation to cloud servers," in *NCCA 2015*. IEEE, 2015.

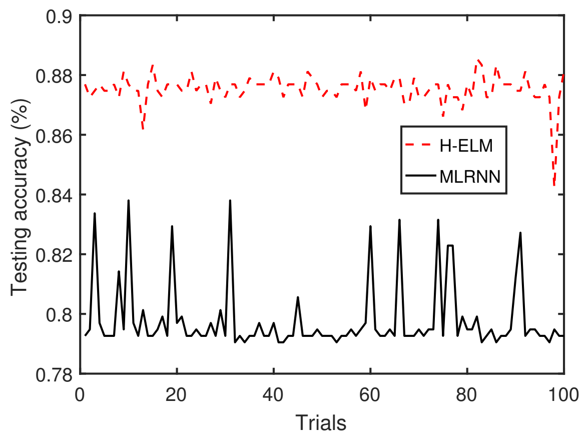

Fig. 6. Testing accuracies of all 100 trials of H-ELM and MLRNN for state detection of Server 3 with Task 2.

[4] D. Simeonov and D. Avresky, "Proactive software rejuvenation based on machine learning techniques," in *International Conference on Cloud Computing*. Springer, 2009, pp. 186–200.

[5] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–14, 2015.

[6] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 575–583, March 2016.

[7] L. Wang, O. Brun, and E. Gelenbe, "Adaptive workload distribution for local and remote clouds," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 003 984–003 988.

[8] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.

[9] ——, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, pp. 154–164, 1993.

[10] E. Gelenbe and C. Cramer, "Oscillatory corticothalamic response to somatosensory input," *Biosystems*, vol. 48, no. 1, pp. 67–75, 1998.

[11] E. Gelenbe and Y. Cao, "Autonomous search for mines," *European J. Oper. Res.*, vol. 108, no. 2, p. 319333, 1998.

[12] E. Gelenbe and T. Koçak, "Area-based results for mine detection," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 38, no. 1, pp. 12–24, 2000.

[13] C. Cramer, E. Gelenbe, and H. Bakircloglu, "Low bit-rate video compression with neural networks and temporal subsampling," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1529–1543, 1996.

[14] C. E. Cramer and E. Gelenbe, "Video quality and traffic qos in learning-based subsampled and receiver-interpolated video sequences," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 2, pp. 150–167, 2000.

[15] H. M. Abdelbaki, K. Hussain, and E. Gelenbe, "A laser intensity image based automatic vehicle classification system," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*. IEEE, 2001, pp. 460–465.

[16] E. Gelenbe, V. Koubi, and F. Pekergin, "Dynamical random neural network approach to the traveling salesman problem," in *Proceedings IEEE Symp. Systems, Man and Cybernetics*. IEEE, 1993, pp. 630–635.

[17] E. Gelenbe and Z. Kazhmaganbetova, "Cognitive packet network for bilateral asymmetric connections," *IEEE Trans. Industrial Informatics*, vol. 10, no. 3, pp. 1717–1725, 2014. [Online]. Available: http://dx.doi.org/10.1109/TII.2014.2321740

[18] E. Gelenbe and F.-J. Wu, "Large scale simulation for human evacuation and rescue," *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3869–3880, 2012.

[19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] R. Raina, A. Madhavan, and A. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proc. 26th Int. Conf. on Machine Learning*. ACM, 2009, pp. 873–880.

[22] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, pp. 3207–3220, 2010.

[23] L. L. C. Kasun, H. Zhou, and G.-B. Huang, "Representational learning with extreme learning machine for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.

[24] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 4, pp. 809–821, 2016.

[25] E. Gelenbe and Y. Yin, "Deep learning with random neural networks," *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1633–1638, 2016.

[26] ——, "Deep learning with random neural networks," *AI Intelligent Systems Conference 2016*, pp. 907–912, 2016.

[27] Y. Yin and E. Gelenbe, "Deep Learning in Multi-Layer Architectures of Dense Nuclei," *ArXiv e-prints*, Sep. 2016.

[28] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[29] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[30] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks." in *Aistats*, vol. 15, no. 106, 2011, p. 275.