J Inf Process Syst, Vol.13, No.2, pp.204~214, April 2017
https://doi.org/10.3745/JIPS.04.0029

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

INVITED PAPER

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Deep Learning in Genomic and Medical Image Data Analysis: Challenges and Approaches

Ning Yu*, Zeng Yu**, Feng Gu***, Tianrui Li**, Xinmin Tian****, and Yi Pan*****

Abstract

Artificial intelligence, especially deep learning technology, is penetrating the majority of research areas, including the field of bioinformatics. However, deep learning has some limitations, such as the complexity of parameter tuning, architecture design, and so forth. In this study, we analyze these issues and challenges in regards to its applications in bioinformatics, particularly genomic analysis and medical image analytics, and give the corresponding approaches and solutions. Although these solutions are mostly rule of thumb, they can effectively handle the issues connected to training learning machines. As such, we explore the tendency of deep learning technology by examining several directions, such as automation, scalability, individuality, mobility, integration, and intelligence warehousing.

Keywords

Bioinformatics, Deep Learning, Deep Neural Networks, DNA Genome Analysis, Image Data Analysis, Machine Learning, lincRNA

1. Introduction

Over the past years, quantum leaps in the quality of a wide range of daily technologies have been doubtlessly noticed. Deep neural networks (DNNs) have been applied to various fields, such as computer vision [1], speech recognition [2], and natural language processing [3]. Studies on the recent advancements in regards to the applications of DNNs have been conducted by Bengio and his colleagues [4-6] and LeCun et al. [7]. Furthermore, the deep learning method has emerged as a state-of-the-art technique for bioinformatics applications [8], such as genomic sequence analysis and medical image analysis. Compared with shallow neural networks, DNNs use enhanced multilayer perceptron (MLP) architecture and complicated layer-wise algorithms to cope with uncertain, imprecise, and approximate problems to achieve robust and tractable outcomes. This represents the learning process of the human brain, and it requires a complex training procedure to learn how to deal with the unknowns of input features.

Manuscript received February 25, 2017; accepted March 21, 2017.
Corresponding Author: Yi Pan (yipan@gsu.edu)
* Dept. of Informatics, University of South Carolina Upstate, Spartanburg, SC 29316, USA (nyu@uscupstate.edu)
** School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China (yuzeng2005@163.com, trli@swjtu.edu.cn)
*** Dept. of Computer Science, College of Staten Island, Staten Island, NY 10314, USA (Feng.Gu@csi.cuny.edu)
**** Intel Compilers and Languages, SSG, Intel Corporation, Santa Clara, CA 95054, USA (xinmin.tan@intel.com)
***** Dept. of Computer Science, Georgia State University, Atlanta, GA 30303, USA (yipan@gsu.edu)

Preliminary ideas about the DNN have been discussed since the 1990s. However, mature concepts about deep learning, including DNNs, were not proposed until the mid-2000s [9-11]. Since then, several works [6,12,13] have applied deep learning in the life sciences and it has showed tremendous promise [14]. Illustrated as examples of DNN applications, the detection of splicing sites of exon/intron and DNA annotation, pattern/motif prediction, and medical image recognition are three of common applications in bioinformatics.

In DNA/gene annotation, splicing sites detection is one example of how DNN methods can be relatively easy to deploy. The splice sites are punctured along DNA sequences where transcription processes rely on these biological marks, and only 1% of dimer AG/GTs are identified as the real splice sites in a DNA sequence [15]. DNNs are superior to the traditional neural network in regards to datasets with a larger number of features through the experiments. However, due to the uncertainty of DNA sequence data, it performs differently in various encoding schemes. Moreover, its tuning process is relatively more difficult than other techniques.

Many tasks in genomic motif/pattern recognition can be performed via deep learning methods [16,17]. For example, in lincRNAs where the complicated bio-chemical mechanisms remain undiscovered can be predicted by DNNs. By scanning the newly found dataset in the RNA-seq, scientists have discovered that: (1) the expression of lincRNAs appears to be regulated, that is, the relevance exists along the DNA sequences; and (2) lincRNAs contain some conversed patterns/motifs that are tethered together by non-conserved regions. These two pieces of evidence provide the reasoning for adopting knowledge-based deep learning methods in lincRNA detection. Driven by those newly annotated lincRNA data, deep learning methods based on auto-encoder algorithm can capture the correlated information along DNA genome sequences by feeding the lincRNA data to the learning machine. DNNs perform better than other techniques, such as the support vector machine (SVM) and traditional neural network. However, it also shows that DNNs have their own issues, such as having more parameters to tune and more complicated architecture-related issues compared with others.

Deep convolution neural networks (DCNNs) perform excellently in image classification. In the framework of image recognition, DCNNs make use of not only image pixels but of the spatial correlation presented in natural images. It has shown that when trained with appropriate regularization [18,19], DCNNs can achieve remarkable performances in visual object recognition tasks. However, through various experiments, it has been shown that architecture is one of the most important factors that can determine the performance in image classification. Better performance may be achieved by some novel neural network structures. It gives scientists lots of spaces to consider what types of structures can have the optimal performance after all. It has shown that constructing and tuning a DNN is not an easy and straightforward task.

Therefore, in this study we summarize the issues in these applications and provide some corresponding solutions. Even though most of them are experience-based approaches, it can effectively handle these issues and make the training of DNNs easier. Based on all of this, we also explore the directions that are developing in the applications of bioinformatics, which are expected to occur in the coming years.

2. Issues and Practical Approaches

Training DNNs is a tougher task to execute than shallow neural networks [20]. When studying these applications, we discovered that the difficulties are caused by the intrinsic characteristics of deep

learning and neural networks. For example, its complicated architecture makes the tuning problem more complex due to more parameters being involved. However, for bioinformatics applications, in addition to the common problems that occur in DNNs, some very specific difficulties come up. For example, data representation in genome analysis becomes prominent over other issues due to the uncertainty of biochemical properties along DNA sequences; data imbalance occurring in medical data leads to the frequent overfitting on training DNNs.

2.1 Data Representation

Data representation is an important part of training mechanism and algorithm design in current deep learning research, such as natural language processing and language translation [21,22]. Similarly, in genome analysis, the letter/character-based representation of a genome sequence is readable and understandable to humans [23] but is a problem for machines, especially for numeric machine learning. We can easily quantify other scientific factors like temperature and humidity and have no problems with man-made quantitative factors like voltage, current, pixels, and coordinates, but there are some problems in quantifying aspects of human biology, such as DNA sequences. No one knows which representation is the best for encoding numeric values in these nucleotides or proteins. However, we cannot avoid using the numeric representations of these biological units when applying learning machine to biological studies. The data representation problems of DNA/protein sequences emerge when the number-based deep learning technology is highlighted.

Inappropriate encoding schemes can directly lead to numeric bias and signal loss. Conventionally, the numeric representations can be summarized into the following three categories: (1) Cartesian coordinate coding, (2) binary linear code, and (3) biochemical mapping. In our previous research, we studied nine encoding schemes, and they typically represent the three categories mentioned above. The nine encoding schemes are DAX [24], arbitrary, EIIP [25], neural [26], complementary [26], enthalpy [27], entropy [28], statistic [28], and Galois [29]. DAX, arbitrary, neural, and Galois are binary linear codes; EIIP, enthalpy, entropy, and statistic are biochemical mapping; and complementary is Cartesian coordinate coding. For DNA genome analysis in DNNs, direct mapping schemes, such as DAX, EIIP, and complementary, are better than pre-processed schemes, such as enthalpy, entropy, and Galois. This might be because direct mapping does not wrap any information from DNA sequences while pre-processed schemes hide some information by encoding it together. Experiments have shown that 'complementary' can beat other schemes in more than half of cases and it is regarded as one of the best encoding schemes in genomic data encoding. Certainly, data encoding is only an aspect of data representation. Other aspects such as data normalization, noisy data processing, and so forth are also important to deep learning in genomic applications.

2.2 Architecture

The architecture of DNNs is one of the fundamental factors in determining the success of training results. It is not only about the number of layers in DNNs but also about the definition and the organization of the layer-layer structure. For example, our previous work on cross-layer neurons demonstrated the critical influence of DNN structures on their performance. Different from the traditional layer-by-layer architecture, they are cross-layer structures where the top layer connects with (controls) all of the lower-level layers, the bottom layer links with (obeys) all of the higher-level layers,

and all the middle layers not only control all of the lower-level layers, but also obey all of the higher-level layers. By using cross-layer architecture, the training results can be improved by around 10%-30% compared to the results from recent layer-by-layer structures [30].

It is more obvious in traditional convolutional neural networks where network structures are constructed layer-by-layer. They have achieved a series of breakthroughs on visual object recognition tasks. However, when these networks become very deep, the information (features) learned by the lower-level layers are dissipated and cannot be sent to the topmost-level layer. In fact, features come from the lower-level layers, such as pixels, edges, texton, and motif, and part of the image can help classify the images into the correct classes.

However, such deep architectures are more difficult to train. Training a deep convolution neural network does not mean simply adding layers. Somehow the whole traditional architectures are overwhelmed. It illustrates that architecture is one of the biggest instability issues that will either boost performance or worsen the case. Based on the architectures, other factors are enabled to consider, such as initialization schemes [31,32], strategies of training [33-35], architectures of networks [36], and so on.

In addition to layer structures, other architecture-related issues include how to find the optimal number of hidden nodes, how to determine the number of layers, how to choose corresponding update functions, etc. Constructing these structure-related factors directly affects the final results. For example, too many hidden-layer nodes and hidden layers can increase the probability of more noise occurring, which can cause the frequent overfitting.

2.3 Hyper-parameters

Once a deep learning architecture is selected, many parameters are subsequently represented to set, including the number of epochs, the number of iterations, the number of mini-batch, and the learning rate, all of which remarkably influence the results [37]. In this era of big data, automation in machine learning research, specially automatically optimizing parameters, has an increasing demand [38]. However, in the past decades, the tuning parameter is not a systematic issue due to the less amount of data and it was left up to those experts on machine learning.

Multiple epochs and iterations are needed for training DNNs. However, multiple iterations are generally used when training small data sets. If one sets too few epochs, not enough time is given to train DNNs and the training results may not be reliable. If too many epochs are set, an overfitting problem might occur during the training process. So far, there is no automatic means to configure the number of epochs. The rule-of-thumb is to monitor the progress and use early stopping, which can stop the training at the early stage and prevent the neural network from overfitting.

A mini-batch represents the pace of computing gradients and parameter updates, namely, the number of samples used at a time. In short, it breaks up data into several mini-batches for gradient calculations and parameter updates. The size of each mini-batch varies. However, we mainly used the range of 16 to 128 for our genomic analysis and bioinformatics applications, which depends on the architecture and other specifications of neural networks. A larger number of mini-batches bigger than 128 in size is proper in some large data sets, and a small number of mini-batches that are below 10 in size are usually too small for parallel acceleration, such as GPUs. Like other parameters, the general rule-of-thumb is still practical for finding the proper number of mini-batches for a particular application.

Learning rate is of less importance among parameters. However, if it is set inappropriately, the learning process may turn very slowly or poorly. Its magnitude is affected by data sets and DNN architectures. The typical values are in the range of 0.1 to 0.000001. In the same vein, tuning the learning rate can be started by trying a few different values (e.g., 0.1, 0.001, and 0.000001). After having a few ideas of what it should be, it can be further tuned. In a distributed environment, different learning rates may be applied as opposed to training on a single machine. The start ratio of updating a parameter for the learning rate can be 0.001.

2.4 Optimization Algorithms and Activation Functions

The most commonly used method for update optimization is stochastic gradient descent (SGD). Although SGD can create instability problems for DNNs, it is widely used because of its simplicity. Layer-wise training for DNNs was created in order to solve the vanishing gradient problem caused by SGD. Alternatively, other optimization algorithms, such as the limited-memory BFGS algorithm, are more powerful than SGD. However, it is criticized for its costly parameter updates. A compromise and practical method in many cases is to combine the SGD with some optimizer algorithms, such as Momentum, AdaGrad, RMSProp, AdaDelta, and Adam [39,40], etc. Using these optimizer algorithms to train DNN results in faster training than when using a traditional SGD. However, it further increases the complexity of the training procedure.

Besides the momentum-based SGD, using the activation function also introduces difficulties to the training process. DNNs have two types of activation functions. One is the activation function for the output layer. The 'softmax' activation function is frequently used in bioinformatics applications. Another type of activation function is for hidden layers where the 'relu' activation function is frequently used. Typical activation functions, such as 'sigmoid' and 'tanh', can lead to vanishing gradient problems, not recommended for training DNNs [20].

2.5 Overfitting

Overfitting in DNNs refers to neural networks fitting the training data sets very well but having poor performance in predicting other new data. Many factors can cause this issue, including imbalanced training data, oversized layers and nodes, inappropriate data representation, the optimization of algorithms and functions, etc. For example, the imbalance of training data is commonly seen in bioinformatics. DNNs require labeled data, while high-throughput medical image data sets are difficult to obtain as many of them are unavailable due to issue of privacy. Moreover, these acquired data sets are not fully annotated because of there being a lack of expertise in the medical domain. On the other hand, rare diseases are not represented in the data sets. The number of samples in the normal class might be remarkably larger than that of samples in the disease class. In many cases, this type of data bias can inevitably result in the network overfitting and negatively affect the performance of DNNs.

Two effective methods to prevent DNNs from overfitting are early stopping and dropout [18], which can help reduce the network's training complexity. This is not a substantial way to solve the problem of overfitting. The essential method is to first research the real reason that causes overfitting and take the appropriate corresponding actions. For example, if it is caused by data imbalance, a separate step may be used to interpret and make up the missing labels; if it is caused by the oversize of network layer, then adjust the number of layers, and so forth.

3. Future Directions

In deep learning research, many problems remain unsolved. Researchers have observed the promising performance of deep learning but cannot give a theoretical explanation as to why the neural network mimicking human brain can bring a better performance. Similarly, the deep learning applications in genomic data and medical image analysis are in the initial stage and have a lot of room to grow and develop. The future development of the technology in these applications probably needs to focus on several directions: automation, scalability, individuality & mobility, and integration & intelligence warehousing. Some terms may literally contradict each other; however, they evolve on the different directions.

3.1 Automation

Automation refers to automating the complicated tuning process, including some of the trends that are listed below.

(1) Adopting statistical methods to optimize the parameters. The automation of parameter settings is an emerging topic in machine learning area. A few algorithms borrowed from statistics have been proposed, including the Bayesian optimization with Gaussian process priors [41,42], sequential model-based global optimization [43], Monto Carlo Metropolis–Hastings algorithm [41], and random search approaches [44].

(2) Exploring the alternative optimization methods of DNNs. Since the main reason of long training time is that parameter optimization through SGD takes too long, several studies have focused on advanced optimization algorithms [45]. Some widely-employed algorithms include AdaGrad, RMSProp, Adam, AdaDelta, batch normalization [19] and Hessian-free optimization [46].

3.2 Scalability

In the context of parallel computing, scalability is the requirement of present development to alleviate the computation complexity and meet the needs of multiple aspects. Parallel and distributed computing can significantly reduce completion time and have enabled many deep learning studies [47-51]. These approaches exploit both scale-up methods, which use GPUs, and distributed methods, such as large-scale clusters, the cloud platform, and a highly-distributed environment. Many parallelization tools, such as CUDA for GPUs [52] and Intel compiler for Intel Xeon Phi coprocessors, can be applied. The recently constructed large-scale deep learning frameworks based on Apache Spark and TensorFlow [53,54] can provide scalable computing capabilities for deep learning applications.

3.3 Individuality and Mobility

Individuality and mobility represent another trend of deep learning technologies. Similarly, in the areas of genomic analysis and medical image analytics, mobility technology and technologies meeting individual needs are the new directions of future development. This means that more specialized and professional services can be provided for individuals. Current models for DNNs are mostly optimized for regular computers rather than for mobility and individuality. However, under the trend of personal genomics and precision medicine cares, mobility and individuality are also inevitable. Some smaller

networks that require less computation are able to run natively in the mobile environment, but it remains quite difficult to make computing programs smaller while retaining accuracy. Thus, it may combine the mobility requirements with common cloud platforms so that the necessary input even training can take place in the individual mobile device while other training can occur in the remote cloud. Current research in these fields is still in the initial stages.

3.4 Integration and Intelligence Warehousing

Integration can occur when different technologies and systems are emerging. In this incoming era of deep learning, technology integration is the indispensable step to integrate resources and bring all artificial intelligence together. A larger computational framework for meeting a variety of needs may emerge in the future development. This type of integration is expected to show in the form of intelligence warehouse, similar to the concept of a data warehouse. That is, intelligence warehousing can integrate a variety of technologies into applications and many intelligence methods can exist in the warehouse in the form of service modules. The key idea is integrating machine learning resources and offering intelligence as a service. From this perspective, the IBM Watson Machine Learning Service is an emerging representative on this direction.

Acknowledgement

This research is supported in part by a NVIDIA Academic Hardware Grant.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[2] L. Deng, M. Seltzer, D. Yu, A. Acero, A. Mohamed, and G. E. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Proceedings of 11th annual conference of the International Speech Communication Association (INTERSPEECH2010)*, Makuhari, Japan, 2010, pp. 1692– 1695.

[3] R. Socher, J. Pennington, E. Huang, A. Ng, and C. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Edinburgh, UK, 2001, pp. 151-161.

[4] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[5] Y. Bengio, "Deep learning of representations: looking forward," in *Proceedings of International Conference on Statistical Language and Speech Processing (SLAP)*, Tarragona, Spain, 2013, pp. 1-37.

[6] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

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

[8] S. Min, B. Lee, and S. Yoon. "Deep learning in bioinformatics," *Briefings in Bioinformatics*, 2016. http://doi.org/10.1093/bib/bbw068.

[9] G. Hinton, P. Dayan, B. Frey, and R. Neal, "The wake-sleep algorithm for unsupervised neural networks," *Science*, vol. 268, no. 5214, pp. 1158-1161, 1995.

[10] G. E. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences*, vol. 11, no. 10, pp. 428-434, 2007.

[11] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, BC, 2013, pp. 8599-8603.

[12] P. Di Lena, K. Nagata, and P. Baldi, "Deep architectures for protein contact map prediction," *Bioinformatics*, vol. 28, no. 19, pp. 2449-2457, 2012.

[13] J. Eickholt and J. Cheng, "Predicting protein residue-residue contacts using deep networks and boosting," *Bioinformatics*, vol. 28, no. 23, pp. 3066-3072, 2012.

[14] M. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey, "Deep learning of the tissue-regulated splicing code," *Bioinformatics*, vol. 30, no. 12, pp. i121-i129, 2014.

[15] C. Trapnell, L. Pachter, and S. L. Salzberg, "TopHat: discovering splice junctions with RNA-Seq," *Bioinformatics*, vol. 25, no. 9, pp. 1105-1111, 2009.

[16] M. K. K. Leung, A. Delong, B. Alipanahi, B. J. Frey, "Machine learning in genomic medicine: a review of computational problems and data sets," *Proceedings of the IEEE*, vol. 104, no. 1, pp 176-197, 2016.

[17] M. W. Libbrecht and W. S. Noble, "Machine learning applications in genetics and genomics," *Nature Reviews Genetics*, vol. 16, no. 6, pp. 321-332, 2015.

[18] 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.

[19] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 448-456.

[20] X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy, 2010, pp. 249-256.

[21] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1724-1734.

[22] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, et al., "Google's neural machine translation system: bridging the gap between human and machine translation," 2016 [Online]. Available: https://arxiv.org/pdf/1609.08144.pdf.

[23] N. Yu, X. Guo, F. Gu, and Y. Pan, "Signalign: an ontology of DNA as signal for comparative gene structure prediction using information-coding-and-processing techniques," *IEEE Transactions on Nanobioscience*, vol. 15, no. 2, pp. 119-130, 2016.

[24] N. Yu, X. Guo, F. Gu, Y. Pan. "DNA AS X: an information-coding-based model to improve the sensitivity in comparative gene analysis," in *Proceedings of International Symposium on Bioinformatics Research and Applications (ISBRA)*, Norfolk, VA, 2015, pp. 366-377.

[25] I. Cosic, "Macromolecular bioactivity: is it resonant interaction between macromolecules? Theory and applications," *IEEE Transactions on Biomedical Engineering*, vol. 41, no. 12, pp. 1101-1114, 1994.

[26] S. B. Arniker, H. K. Kwan, N. F. Law and D. P. K. Lun, "DNA numerical representation and neural network based human promoter prediction system," in *Proceedings of 2011 Annual IEEE India Conference (INDICON)*, Hyderabad, India, 2011, pp. 1-4.

[27] G. Kauer and H. Blocker, "Applying signal theory to the analysis of biomolecules," *Bioinformatics*, vol. 19, no. 16, pp. 2016-2021, 2003.

[28] K. Jabbari and G. Bernardi, "Cytosine methylation and CpG, TpG (CpA) and TpA frequencies," *Gene*, vol. 333, pp. 143-149, 2004.

[29] G. L. Rosen, "Examining coding structure and redundancy in DNA," *IEEE Engineering in Medicine and Biology Magazine*, vol. 25, no. 1, pp. 62-68, 2006.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016, pp. 770-778.

[31] D. Mishkin and J. Matas, "All you need is a good init," in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016, pp. 1-13.

[32] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, GA, 2013, pp. 1139-1147.

[33] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *Journal of Machine Learning Research*, vol. 10, pp. 1-40, 2009.

[34] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, GA, 2013, pp. 1310-1318.

[35] M. Bianchini, and F. Scarselli, "On the complexity of shallow and deep neural network classifiers," in *Proceedings of 22nd European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2014, pp. 371-376.

[36] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-ResNet and the impact of residual connections on learning," 2016 [Online]. Available: https://arxiv.org/pdf/1602.07261.pdf.

[37] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade.* Heidelberg: Springer, 2012, pp. 437-478.

[38] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, "Algorithms for hyper-parameter optimization," *Advances in Neural Information Processing Systems*, vol. 24, pp. 2546-2554, 2011.

[39] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121-2159, 2011.

[40] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014 [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf.

[41] J. F. G. de Freitas, "Bayesian methods for neural networks," Ph.D. dissertation, University of Cambridge, UK, 2000.

[42] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, pp. 2951-2959, 2012.

[43] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*. Heidelberg: Springer, 2011, pp. 507-523.

[44] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281-305, 2012.

[45] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, WA, 2011, pp. 265-272.

[46] J. Martens, "Deep learning via Hessian-free optimization," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 735-742.

[47] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, Montreal, Canada, 2009, pp. 873-880.

[48] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," *Advances in Neural Information Processing Systems*, vol. 26, pp. 1223-1231, 2013.

[49] Y. Bengio, H. Schwenk, J. S. Senecal, F. Morin, and J. L. Gauvain, "Neural probabilistic language models," in *Innovations in Machine Learning*. Heidelberg: Springer, 2006, pp. 137-186.

[50] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B. Y. Su, "Scaling distributed machine learning with the parameter server," in *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Broomfield, CO, 2014, pp. 583-598.

[51] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, et al., "Large scale distributed deep networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1223-1231, 2012.

[52] C. Angermueller, T. Parnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Molecular Systems Biology*, vol. 12, article no. 878, pp. 1-16, 2016.

[53] H. Kin, J. Park, J. Jang, and S. Yoon, "DeepSpark: spark-based deep learning supporting asynchronous updates and Caffe compatibility," 2016 [Online]. Available: https://arxiv.org/pdf/1602.08191.pdf.

[54] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, et al., "Tensorflow: large-scale machine learning on heterogeneous distributed systems," 2016 [Online]. Available: https://arxiv.org/pdf/1603.04467.pdf.

Ning Yu http://orcid.org/0000-0001-8861-0804

He is currently an assistant professor of Informatics in University of South Carolina Upstate. He received his Ph.D. degree in Computer Science in Georgia State University. He received M.Sc. degree in Computer Science from Southern Illinois University Carbondale in 2009. His B.Sc. degree in Computer Science and M.Eng. degree in Signal Processing were received from Communication University of China. His research area includes Data Mining, Deep Learning, Bioinformatics, Parallel and Cloud Computing.

Zeng Yu http://orcid.org/0000-0003-3550-3495

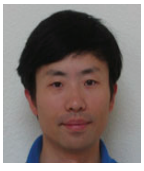
He received B.S. and M.S. degrees from the Department of Mathematics, School of Sciences, China University of Mining and Technology in 2008 and 2011, respectively. He is currently a PhD candidate in the School of Information Science and Technology, Southwest Jiaotong University, China. He is also a visiting PhD student in the Department of Computer Science, Georgia State University, USA. His current research interests include data mining, bioinformatics, deep learning and cloud computing.

Feng Gu http://orcid.org/0000-0001-5337-4282

He received his B.Sc. degree in mechanical engineering from China University of Mining and Technology in 1998 and M.Sc. degree in information systems from Beijing Institute of Machinery in 2003. He received his M.Sc. and Ph.D. in computer science from Georgia State University in 2009 and 2011, respectively. He was an assistant professor of computer science at Voorhees College from 2010 to 2013. He is currently an assistant professor of Computer Science at College of Staten Island, The City University of New York, and the doctoral faculty member of Graduate Center of The City University of New York. He is the recipient of Natural Science Foundation Research Initiation Award. His research interests include modeling and simulation, complex systems, high performance computing, and bioinformatics.

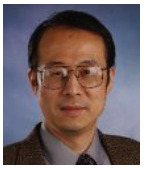
Tianrui Li

He received the B.S., M.S., and Ph.D. degrees from the Southwest Jiaotong University, Chengdu, China, in 1992, 1995, and 2002, respectively. He was a postdoctoral researcher with SCK•CEN, Belgium, from 2005 to 2006, and a visiting professor with Hasselt University, Belgium, in 2008, the University of Technology, Sydney, Australia, in 2009, and the University of Regina, Canada, 2014. He is currently a professor and the director of the Key Laboratory of Cloud Computing and Intelligent Techniques, Southwest Jiaotong University. He has authored or coauthored more than 150 research papers in refereed journals and conferences. His research interests include big data, cloud computing, data mining, granular computing, and rough sets. He is a fellow of IRSS and a senior member of the ACM and IEEE.



Xinmin Tian

He is a Senior Principal Engineer and Compiler Architect at Intel, who drives compiler parallelization/OpenMP, vectorization, offloading and optimization technologies for Intel multi-core and many core architectures, programming language extensions and tools in multiple domains. He serves as the Intel ARB representative at OpenMP Architecture Review Board (ARB). He holds 25+ US Patents, published 50+ technical papers and is a co-author for two books on compiler optimizations, parallel programming and performance tuning.



Yi Pan http://orcid.org/0000-0002-2766-3096

He is a Regents' Professor of Computer Science and an Interim Associate Dean at Georgia State University, USA. Dr. Pan received his B.Eng. and M.Eng. degrees in computer engineering from Tsinghua University, China, in 1982 and 1984, respectively, and his Ph.D. degree in computer science from the University of Pittsburgh, USA, in 1991. His profile has been featured as a distinguished alumnus in both Tsinghua Alumni Newsletter and University of Pittsburgh CS Alumni Newsletter. Dr. Pan's research interests include parallel and cloud computing, wireless networks, and bioinformatics. Dr. Pan has published more than 150 journal papers with over 50 papers published in various IEEE journals. In addition, he has published over 150 papers in refereed conferences. He has also co-authored/co-edited 37 books. His work has been cited more than 4,000 times. Dr. Pan has served as an editor-in-chief or editorial board member for 15 journals including 7 IEEE Transactions. He is the recipient of many awards including IEEE Transactions Best Paper Award, IBM Faculty Award, JSPS Senior Invitation Fellowship, IEEE BIBE Outstanding Achievement Award, NSF Research Opportunity Award, and AFOSR Summer Faculty Research Fellowship. He has organized many international conferences and delivered over 40 keynote speeches at various international conferences around the world.