et al., 2009). Extending the model to investigate oscillatory and other more intricate GTPase spatio-temporal dynamics could be an important next step in the link between theory and experiments, promising to shed more light on the central regulatory units of the cell.

REFERENCES

Byrne, K.M., Monsefi, N., Dawson, J.C., Degasperi, A., Bukowski-Wills, J.-C., Volinsky, N., Dobrzyński, M., Birtwistle, M.R., Tsyganov, M.A., Kiyatkin, A., et al. (2016). Cell Syst. *2*, this issue, 38–48.

Guilluy, C., Garcia-Mata, R., and Burridge, K. (2011). Trends Cell Biol. *21*, 718–726.

Machacek, M., Hodgson, L., Welch, C., Elliott, H., Pertz, O., Nalbant, P., Abell, A., Johnson, G.L., Hahn, K.M., and Danuser, G. (2009). Nature *461*, 99–103.

Nguyen, L.K., Degasperi, A., Cotter, P., and Kholodenko, B.N. (2015). Sci. Rep. *5*, 12569.

Nobes, C.D., and Hall, A. (1995). Cell *81*, 53–62.

Ridley, A.J., Paterson, H.F., Johnston, C.L., Diekmann, D., and Hall, A. (1992). Cell *70*, 401–410.

Symons, M., and Segall, J.E. (2009). Genome Biol. *10*, 213.

Tsyganov, M.A., Kolch, W., and Kholodenko, B.N. (2012). Mol. Biosyst. *8*, 730–743.

Wang, Y., Ku, C.J., Zhang, E.R., Artyukhin, A.B., Weiner, O.D., Wu, L.F., and Altschuler, S.J. (2013). Cell Rep. *3*, 1607–1616.

Xu, J., Wang, F., Van Keymeulen, A., Herzmark, P., Straight, A., Kelly, K., Takuwa, Y., Sugimoto, N., Mitchison, T., and Bourne, H.R. (2003). Cell *114*, 201–214.

TensorFlow: Biology's Gateway to Deep Learning?

Ladislav Rampasek[1,2] and Anna Goldenberg[1,2,*]
[1]SickKids Research Institute, 686 Bay Street, Toronto, ON M5G 0A4, Canada
[2]Department of Computer Science, University of Toronto, 40 St. George Street, Toronto, ON M5S 2E4, Canada
*Correspondence: anna.goldenberg@utoronto.ca
http://dx.doi.org/10.1016/j.cels.2016.01.009

TensorFlow is Google's recently released open-source software for deep learning. What are its applications for computational biology?

Machine learning, particularly the flavor known as deep learning, powers Google's ability to decipher spoken search queries and to recognize your face in a photo. New algorithms, massively parallel computational hardware, and landmark successes, such as surpassing human-level accuracy in object recognition from images, have made deep learning one of the hottest topics in computer science in recent years (reviewed in LeCun et al., 2015), with researchers at Google at the forefront of methods development. By releasing their in-house-developed deep learning framework TensorFlow as open-source software (Abadi et al., 2015), Google has provided a stable platform for deep learning research and applications. The possibilities for TensorFlow in biological research are tantalizing, at the very least offering computational biologists a more standardized system that is likely to improve ease of use and reproducibility of deep learning methods.

A deep learning model, typically a multi-layer neural network, is composed of several computational layers that process data in a hierarchical fashion. Each layer takes an input and produces an output, often computed as a non-linear function of a weighted linear combination of the input values. The output of one layer becomes an input to the next processing layer, creating a deep architecture (Figure 1). In each successive layer, the data are being represented in an increasingly more abstract way. Particularly popular are "convolutional layers" that apply a local function, called a filter, to all subsets of the layer's input, such as portions of an image. For example, in an object recognition task, the first layers represent basic features like lines, curves, and other visual primitives. Filters recognizing such visual primitives are learned by the model, not hard coded. The higher layers encode for presence of more complex shapes, for example, a human face. Such deep models can learn a highly complex data abstraction that captures the intricate structure of very large datasets.

In computational biology, the field of protein structure prediction was one of the earliest to recognize the potential of neural networks, applying them since the 1980s. It was also one of the early adopters of deep learning. For example, a 50-layer deep learning model improved contact map predictions on the CASP8 dataset by 10%—a significant boost relative to previous advances (Di Lena et al., 2012). Deep learning received more widespread attention in computational biology after the 2012 MERCK Molecular Activity Challenge, when a deep model won the competition (Dahl et al., 2014). What is interesting is that the winning team had purely machine learning expertise and no domain knowledge. Deep learning has also been successful in medical imaging applications, such as segmenting brain images to help diagnose Alzheimer's disease and determining correspondence between images (image registration).

Most recently, methods based on deep learning became the state-of-the-art in predicting various regulatory effects directly from DNA sequences. Xiong et al. (2015) predicted the effect of single

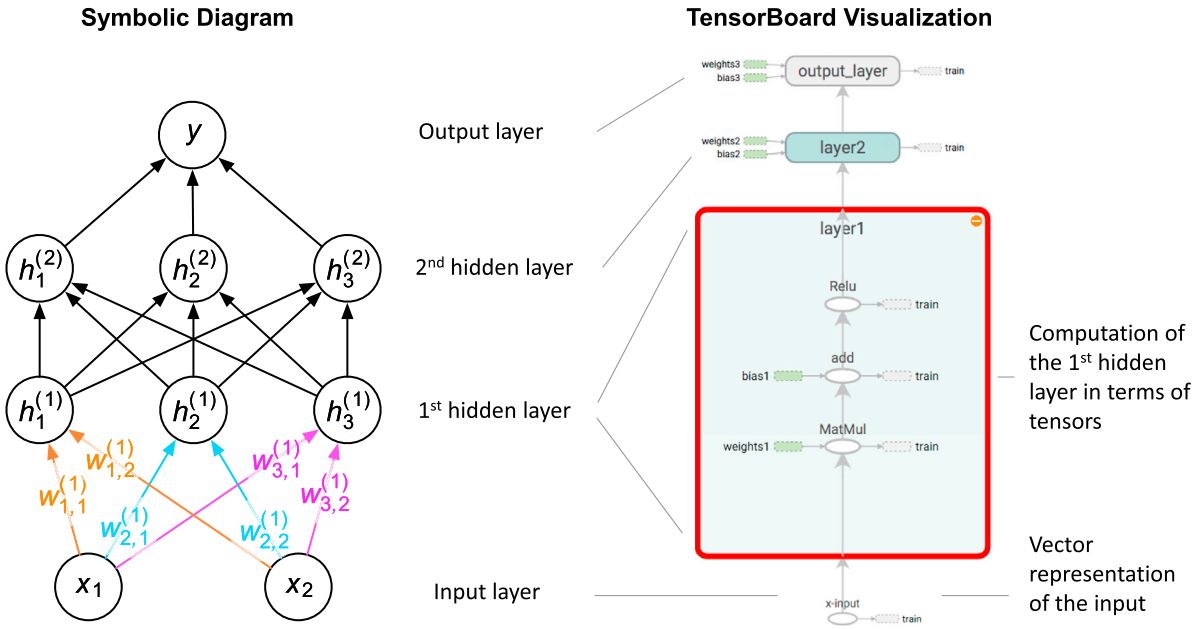

Figure 1. An Example of a Feed-Forward Neural Network with Two Hidden Layers

The left panel depicts a diagram of the neural network. The right panel shows the corresponding TensorFlow computational graph as a screenshot of the TensorBoard tool, the graphical part of the TensorFlow package. This neural network takes a vector $\boldsymbol{x} = (x_1, x_2)$ as the input. The values (units) in the first hidden layer are computed as a non-linear function of a weighted linear combination of the inputs, e.g. $h_1^{(1)} = max(0,\ w_{1,1}^{(1)}x_1 + w_{1,2}^{(1)}x_2 + b_1^{(1)})$. In the TensorBoard visualization, this computation is expressed in terms of matrices (tensors) as $\boldsymbol{h}^{(1)} = max(\boldsymbol{0},\ W^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)})$. The non-linearity used here is the so-called rectifier function, $Relu(x) = max(0, x)$. The second hidden layer is computed analogously. Note that, in practice, neural networks can consist of many layers of different types (such as convolutional or pooling layers) with several thousand units each. The visualization of the graph structure in TensorBoard is interactive, enabling both compact and detailed representation of the underlying computations. Here, details of the computation in the first hidden layer are shown, whereas details of the computations in the other layers or functions are compressed and shown as single nodes.

point mutations within introns or exons on RNA splicing (including variants far from splice site). Alipanahi et al., (2015) introduced DeepBind, a convolutional neural network predicting sequence specificities of DNA- and RNA-binding proteins. Similarly, DeepSEA used deep learning to infer regulatory sequence code from chromatin-profiling data (Zhou and Troyanskaya, 2015).

Implementing a successful complex deep neural network model is a non-trivial task and has been essentially restricted to the deep learning experts or those that became such in the process. Fortunately, thanks to the modular structure of the networks and standard inference tools, several software frameworks that speed up the design and training of deep neural networks are now available.

TensorFlow is the newest addition to this toolbox. It provides several improvements, such as graphical visualization and improved compilation time. The deep learning frameworks most widely used today are Torch7, Theano, and Caffe, which is particularly suitable for convolutional neural networks (Table 1). Additionally, several start-up companies have also open sourced their deep learning tools—for example, neon (a Python-based framework by Nervana), Deeplearning4J (a Java tool by Skymind), and H2O-3 (a Java-based machine learning toolkit with interfaces to Python, R, Scala and others, developed by H2O).

TensorFlow, like Theano, uses a declarative programming paradigm. This allows researchers to focus on the symbolic definition of what needs to be computed rather than how exactly, and in what particular order, these computations are to be performed, which is the case in imperative programming. The computational model, a deep neural network, is then represented by a symbolic computational graph (Figure 1). This abstract representation of the model can be optimized for numerical stability and performance, and individual parts can be translated to be executed by a computer processor or graphics chip. Perhaps most importantly, the symbolic representation allows for automatic differentiation, which provides a convenient way to optimize many functions. These can be neural networks or other functions that are commonly used to mathematically represent different data-driven problems.

Processing the abstract representation of the model is handled automatically by the framework. This makes TensorFlow and Theano especially suitable for development of novel models that use gradient-based optimization. Deep neural network architectures, but also other kinds of models, fall into this category. The major drawback of Theano is the time it takes to compile the symbolic model. TensorFlow significantly improves upon this bottleneck. Another advantage of TensorFlow is that it comes with a supporting tool named TensorBoard for in-depth visualization of the model training progress. One can interactively investigate the structure of the computational graph, as well as how the parameters and model performance are changing over the training iterations (Abadi et al., 2015).

Table 1. Comparison of TensorFlow with the Most Popular Open-Source Deep Learning Frameworks

Framework	Core Programming Language	Interfaces from Other Languages	Programming Paradigm	Wrappers
Caffe[a]	C++/CUDA	Python, Matlab	Imperative	
TensorFlow[b]	C++/CUDA	Python	Declarative	Pretty Tensor, Keras
Theano[c]	Python (compiled to C++/CUDA)	–	Declarative	Keras, Lasagne, or Blocks
Torch7[d]	LuaJIT (with C/CUDA backend)	C	Imperative	

[a](Jia et al., 2014). Developed by the Vision and Learning Center at UC Berkeley.
[b](Abadi et al., 2015). Developed by Google.
[c](Bastien et al., 2012). Developed by Université de Montréal.
[d](Collobert et al., 2011). Developed by Facebook, Google, Twitter, New York University, and others.

TensorBoard visualization provides a modular representation of a very complex model. This facilitates global representation of the model, debugging, and model checking to gain insight during development of the model (Figure 1).

At the time of writing, the open source version of TensorFlow can run only on a single machine. However, it supports parallelization over multiple processors (CPUs or GPUs) of the single machine. A version of TensorFlow capable of distributed computation on a cluster of machines is slated for release later in 2016. This is a standard progression of deep learning tools: the majority of the current frameworks do not support distributed computation out of the box either. After the initial release of TensorFlow in November 2015, multiple concerns, such as run time and extensive memory usage, were raised as possible causes of the underwhelming performance of TensorFlow compared to other state-of-the-art deep learning frameworks. Google has acknowledged these initial performance issues, and in a new version 0.6 released in December, many of these have been addressed. This fast turnaround indicates Google's commitment to supporting and developing this open source project.

Is there anything to be gained from TensorFlow by the computational biology community? We believe there is. Streamlined graphics, faster compilation time, and Google's claim that there is nothing preventing an implementation of a front end in the R programming framework if the community desires are all welcome additions to the suite of deep learning frameworks. On the downside, TensorFlow, like Theano, is still a low-level framework. In other words, users will have to know how to program and understand the necessary concepts of neural network model design and training. Here, high level wrappers of TensorFlow (Table 1), such as Keras or Pretty Tensor, help with these tasks by providing a simplified way to build neural networks from standard types of layers. Importantly, having the backing and commitment of Google, which is on the forefront of deep learning model development, provides hope that TensorFlow will facilitate a standardized and thus reproducible platform for developing deep learning approaches in computational biology.

There is a more important question that needs to be answered before any of these frameworks should be considered. When is deep learning applicable in computational and systems biology in general? One prerequisite for using current deep learning approaches is a dataset with many samples. Most cohort-based studies where the goal is, for example, to identify genes or methylation probes associated with a given disease have a small sample size and thus cannot be analyzed with this powerful technology. The trick to using deep learning models is to think of clever ways to represent the data where each event, such as splicing, RNA-protein binding, or methylation, is a training sample in your data, thus creating a scenario where the number of samples far exceeds the number of variables. Once this crucial step is achieved, TensorFlow provides a powerful and flexible gateway to play with deep learning.

ACKNOWLEDGMENTS

Authors are not affiliated with Google Inc. and declare no conflict of interest. TensorFlow, the TensorFlow logo, and any related marks are trademarks of Google Inc. This work was supported by the SickKids Research Institute (A.G.).

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. http://download.tensorflow.org/paper/whitepaper2015.pdf.

Alipanahi, B., Delong, A., Weirauch, M.T., and Frey, B.J. (2015). Nat. Biotechnol. *33*, 831–838.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements. arXiv, arXiv:1211.5590, http://arxiv.org/abs/1211.5590.

Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. *BigLearn, NIPS Workshop*.

Dahl, G.E., Jaitly, N., and Salakhutdinov, R. (2014). Multi-task neural networks for QSAR predictions. arXiv, arXiv:1406.1231, http://arxiv.org/abs/1406.1231.

Di Lena, P., Nagata, K., and Baldi, P. (2012). Bioinformatics *28*, 2449–2457.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. Proceedings of the ACM International Conference on Multimedia. ACM, pp. 675–678.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Nature *521*, 436–444.

Xiong, H.Y., Alipanahi, B., Lee, L.J., Bretschneider, H., Merico, D., Yuen, R.K., Hua, Y., Gueroussov, S., Najafabadi, H.S., Hughes, T.R., et al. (2015). Science *347*, 1254806.

Zhou, J., and Troyanskaya, O.G. (2015). Nat. Methods *12*, 931–934.