Visualization System for Evolutionary Neural Networks for Deep Learning

Junghoon Chae, Catherine D. Schuman, Steven R. Young, J. Travis Johnston, Derek C. Rose, Robert M. Patton and Thomas E. Potok
Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA
{chaej, schumancd, youngsr, johnstonjt, rosedc, pattonrm, potokte}@ornl.gov

***Abstract*—Deep learning is actively used in a wide range of fields for scientific discovery. To effectively apply deep learning to a particular problem, it is important to select an appropriate network architecture and other hyper-parameters (at each layer). Evolving architectures and hyper-parameters using a genetic algorithm is one current approach to search the huge space of all possible configurations to find those more optimal for the problem. However, examining an evolutionary process and tuning the genetic algorithm are challenging, pushing most users to treat the process as a black box. To address this challenge, we propose a visualization system for evolutionary neural networks for deep learning. The key feature of our visualization system is to provide a visual analytics environment for evaluating a genetic algorithm in order to improve the underlying operations to reduce time to find good solutions. Our system is able to not only visualize how a genetic algorithm traverses its search space but also allows users to examine evolving networks in-depth to get insights to improve performance through interactive visualization components.**

***Index Terms*—visualization, neural network, evolutionary algorithm**

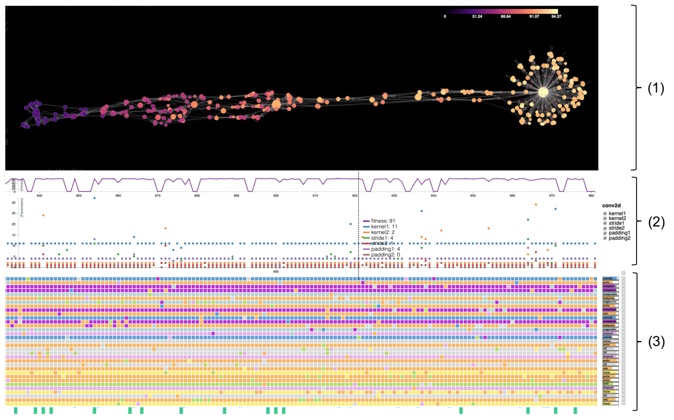

Fig. 1. System Overview: The system consists of three different views: a lineage view, a fitness-parameter view, and a network architecture view.

I. Introduction

Neural networks for deep learning are actively used for scientific discovery in a wide range of fields, such as materials science, physics, and biomedical applications. Unfortunately, there is no one neural network model that can solve many different problems and work for different types of data. Instead, for solving a particular problem, we should select an appropriate network architecture and a set of hyper-parameters for each layer. Evolving neural networks is an efficient approach to find appropriate model configurations in the huge search space. In evolutionary computation, genetic algorithms and evolutionary strategies are capable of training neural networks with a huge number of parameters [1]–[3]. However, the evolutionary process—how crossover and mutation operations assemble the final network—is treated as a black box. Also, examining and tuning the genetic algorithm parameters are also challenging.

Notice: This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Visualization for a genetic algorithm can be helpful to open the underlying mechanics of the genetic algorithm and support the tuning step [4], [5]. In this paper, we propose a visualization system for evolutionary neural networks for deep learning. Our visualization system provides a visual analytics environment for evaluating a genetic algorithm in order to improve the genetic algorithm to search for the optimized solution in less time. Our system is able to not only visualize how a genetic algorithm traverses a search space, but also allows the users to examine evolving neural networks in-depth and get insights to improve its performance through many interactive visualization components. Our visualization system consists of three major views: a lineage view, a fitness-parameter view, and a network architecture view. The three views visualize different features of the evolutionary process. This allows the results of evolutionary computation to be visualized from different perspectives. Also, the views are connected to each other and provide interactive analysis features. The contributions we focus on include:

- Our system helps users analyze how a genetic algorithm explores the search space.
- Our system allows users to see convergence behavior—how a genetic algorithm finds solutions (networks) for a particular problem.
- Our system allows users to analyze the relevance of

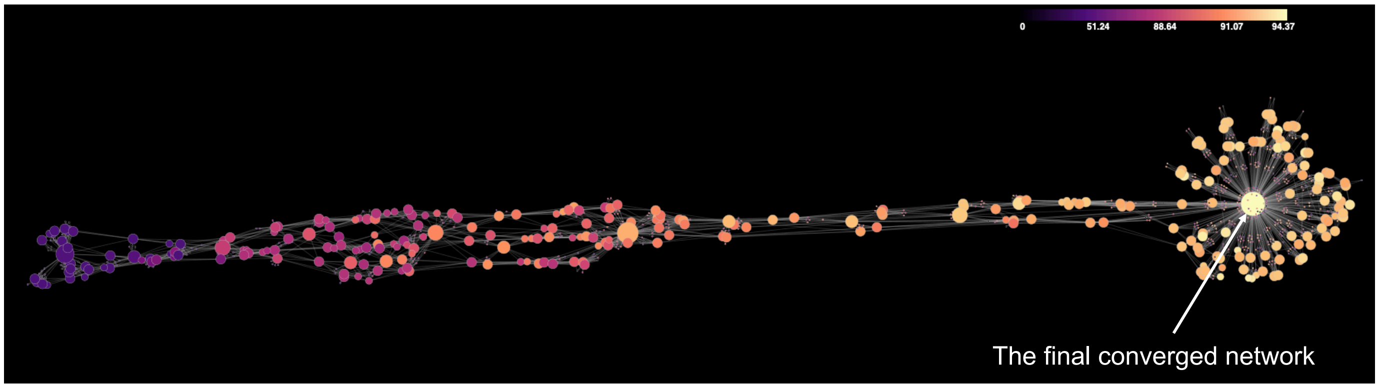

Fig. 2. Lineage View: this view uses a force-directed layout to show parent-child relationships of networks of the evolutionary process. The size of node represents the number of child nodes(networks) and node colors indicate fitness of nodes.

multiple dimensions over the evolutionary process.

- Our system shows how the network architectures converged as evolution progresses.

II. Background

Visualization of Genetic Algorithm: A genetic algorithm is based on the biological methods to produce the best solutions for a particular problem [6]. These algorithms are based on random search methods and widely used to solve optimization problems. However, understanding and improving the searching process of a genetic algorithm are challenging. Many studies have tackled the challenges using visualization-based approaches [4], [5], [7]. The previous studies have focus on showing static images of ancestry trees or traditional graphs and visualize few dimensions of the search space of the genetic algorithm. Farooq et al. [8] proposed an interactive interface to interrupt the searching process, though it has still many limitations to understand the evolutionary process. Our visualization system provides highly interactive analytics features for more comprehensive analysis of evolutionary process. Our system gives users the ability to explore entire dimensions of the search space as well as analyze the connections between multiple dimensions.

Evolutionary Neural Network: When applying deep learning techniques to a specific problem, a necessary step is finding an appropriate neural network model for the problem. Finding an appropriate model includes selecting an appropriate network architecture and a set of hyper-parameters for each layer. This step is usually time-consuming and requires much trial and error. Evolving neural networks using a genetic algorithm is an efficient approach to find a good network configuration in the huge search space [1], [9], [10]. The evolutionary approach taken by Multi-node Evolutionary Neural Networks for Deep Learning (MENNDL) [1] encodes an individual–a single neural network–as a sequence of genes; the genes provide instructions for how to (uniquely) reconstruct the neural network. MENNDL uses HPC resources with many GPUs (such as the Summit supercomputer) to train many different individuals simultaneously, one individual per GPU. The fitness of an individual can be scored in many ways but the focus is typically on the accuracy of the trained neural network on a validation set. As individuals from the population of networks are trained and evaluated, the best individuals (those with the highest fitness) become parents to the next generation of individuals. A child is created by performing cross-over and mutation on the genes of its parents and is subsequently evaluated in a similar fashion. The process repeats (asynchronously) until the genetic algorithm converges, hopefully to a network with sufficiently high fitness. During the evolutionary process we store the characteristics of individuals (fitness, accuracy, and other features) as well as the lineage of the individuals (i.e. which networks were the parents of this network).

III. System Overview and Data

The visualization system we propose contains three views that visualize different aspects of the networks in the evolutionary process of MENNDL (see Figure 1). The first view is a lineage view, showing parent-child relationships in the evolution, as shown in Figure 1 (1). In this view, we utilize a forced-directed layout to show the lineage, where nodes represent parent/child networks and edges represent their relationships. The second view is a fitness-parameter view, showing the changes in fitness and hyper-parameters of the networks as evolution progresses using line and scatter plots, as shown in Figure 1 (2). The third view is a network architecture view, which shows how the layers of each network are stacked and how the network architectures converged as evolution progresses, as shown in Figure 1 (3).

The views are tightly connected each other and provide interactive user interfaces for analysis. The system allows users to link over the three views in order to understand the eovolutionary process better through interacting with the presented data through different representations. The interaction techniques of the views, such as exploration, filter, selection, and reconfigure, overcome the limitation of static representations and support to uncover insights [11].

Data for our system is generated by the MENNDL framework. However, as the format of data is general and simple, making any framework generate data for the system is easy. The system uses two different types of data: lineage data and

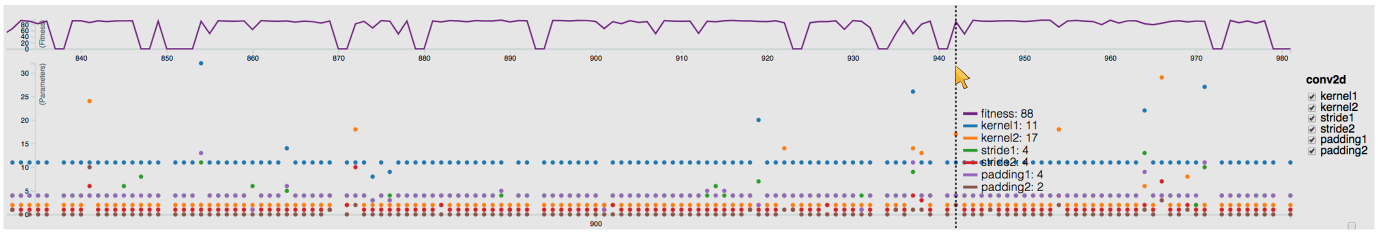

Fig. 3. Fitness-Parameter View: Line plot for fitness (Top) and Scatter plot for hyper-parameters (Bottom). The two plots are vertically aligned. Users are able to interactively explore and examine fitness and the hyper-parameters.

network data. The lineage data contains the lineage information of networks, i.e., parent-child relationships. The network data contains the network topology and hyper-parameters of each layer. The format of the network data we use is prototxt or json. The two data files are connected by sharing unique network IDs.

IV. Lineage View

The central idea of the lineage view is to provide an immediate intuitive understanding of the dynamics of the whole evolutionary process by combining other features of evolutionary computation. The dynamics of evolution are represented as an ancestry tree while we integrate other features into the tree. Note that the ancestry tree generated by the genetic algorithm of MENNDL does not have clearly distinguished generations due to its asynchronous nature. Under MENNDL's asynchronous evolution, child networks can be mutated or recombined from any ancestor networks that already exist. As such, the traditional tree diagram is not appropriate to show the complex lineage information. We utilize a force-directed layout [12]. The force-directed layer is one of the best choices to show the complete ancestry of the networks. The force-directed layout is able to capture the relations between parent and child networks even though there is no explicit generations in the evolutionary process. In this view, each node represents a network and the edges show the relationships between the networks, as shown in Figure 2.

The size of each node represents the number of children of the node. This means that the larger node has more influence on the entire ancestry tree. The color of the node indicates the fitness value of the corresponding network. Networks with low fitness are dark purple and networks with high fitness are bright yellow. We let the force simulator make all nodes repel against each other. However, the link force pushes linked nodes together. Eventually, the nodes that have a direct parent-child relationship are adjacently placed. This allows users to see the changing characteristics of populations as evolution progresses even though there are no explicit generations. For example, Figure 2 shows how fitness and lineage change. Note the large bright yellow node at the right hand side of Figure 2. This node is the final converged network for the genetic algorithm which has the highest fitness score and the largest number of child networks. In this view, when the mouse cursor hovers over a specific node, detailed information for that node is shown. In result, this visualization system allows users to analyze how a genetic algorithm explores the search space and discover the path taken by the genetic algorithm to find solutions for a particular problem.

V. Fitness-Parameter View

The fitness-parameter view allows the users to interactively explore and compare how fitness and selected hyper-parameter values of networks change over the course of evolution. As mentioned earlier, we need an efficient method to find an appropriate set of hyper-parameters in searching a high-dimensional hyper-parameter space. When we use the approach based on evolution, an analogy between fitness and hyper-parameters is helpful in finding appropriate hyper-parameter sets [7], [13]. For example, if we realize specific situations that cause fitness decreases, we can tune the genetic algorithm to avoid such situations. Also, if we find what could be unimportant dimensions of the hyper-parameter space through the visualization, we can also adjust the algorithm to skip the unimportant dimensions.

The fitness-parameter view consists of two parts: a line plot for fitness (Top) and a scatter plot for hyper-parameters (Bottom), as shown in Figure 3. After users select a specific layer in the network architecture view, the scatter plot appears and shows the hyper-parameters of the selected layer (described more in detail in Section VI). The x-axis of the two plots is the index of networks ordered according to the creation time. The y-axis of the line plot is the fitness value, and the y-axis of the scatter plot is the parameter value according to the selected layer and parameters. The two plots are aligned vertically. when the mouse cursor hovers over the view, a dashed vertical line and detailed values appear and users can easily compare the values (See Figure 3). Each parameter has different color. The user can select and view the parameters of interest using the check boxes on the right side of the view.

VI. Network Architecture View

The major goal of the network architecture view is to explore evolution of the network architectures—how the layers of each network are arranged and how the network architectures are converged over evolution progress. In this view, users are able to focus on network architectures rather the hyper-parameters for layers. As we described in Section II, the search space of the genetic algorithm of MENNDL includes the arrangement of layers within a network as well. The network

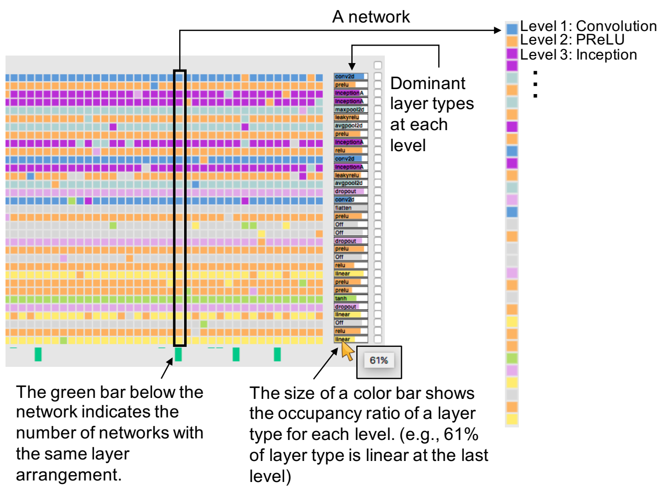
A network
Level 1: Convolution
Level 2: PReLU
Level 3: Inception
Dominant layer types at each level
61%
The green bar below the network indicates the number of networks with the same layer arrangement.
The size of a color bar shows the occupancy ratio of a layer type for each level. (e.g., 61% of layer type is linear at the last level)

Fig. 4. Network Architecture View: A set of vertically aligned small boxes represent a network architecture. The series of the box sets show the evolution of the network architectures.

architecture is one of the most important aspects that affect the performance of convolutional neural network models [9], [10].

In this view, a network architecture is represented by a set of small boxes that are vertically aligned in Figure 4 (Right). Each box represents a layer and the arrangement of boxes reflects the arrangement of the layers in the network. For example, the top box in the set means the layer at the first level and the layer is a convolution layer. The colors of boxes represent specific layer types, and we use the same color for layers with similar types. For example, relu, prelu, and leakyrelu layers have a same orange color. The sets of boxes are listed horizontally in order by the corresponding network creation time. In the example shown in Figure 4 (Left), users can easily perceive how the network architectures evolve. Also, when an user clicks a node in the lineage view, the corresponding network is highlighted. Visualizing the connections between the multiple features (lineage, fitness and architecture) helps the user analyze the performance of the genetic algorithm. There are green bars below each set of boxes. The size of a green bar indicates the number of networks with the same layer arrangement for the corresponding network. For example, if a bar below a network is long, there are many networks that have the identical layer arrangement.

At the right side of view, there are color bars at each level in Figure 4. The colors of each bar show dominant layer types for each level and the bar size represents the occupancy ratio of the dominant layer type. Also, users can see the layer type texts inside the boxes and the ratio number appears when a mouse cursor hovers over a bar. This visualization component gives the summary of the network architectures of the whole evolutionary process. The information can be useful for tuning the genetic algorithm. Also, when users click one of the bars, the hyper-parameters of the corresponding layer appear on the fitness-parameter view. Figure 3 shows the

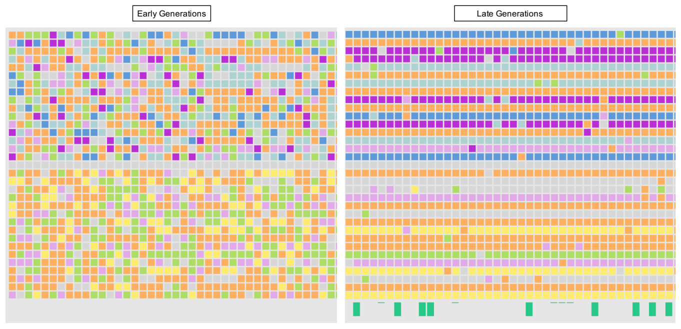

Fig. 5. Comparison of network architectures between early (Left) and late (Right) generations: There are no specific patterns in the early generation but there is a specific converged pattern in the late generation.

example of selection of the convolution layer at the first level. The users can easily compare corresponding hyper-parameters and the network architecture because the fitness-parameter and network architecture views are also vertically aligned (See Figure 1).

As a case study, the two images in Figure 5, show two sub-sets of the entire network architectures. The left image represents an early generation (network index: 0 to 40) and the right one represents a late generation (network index: 800 to 840, the final index is 980). We can clearly perceive the difference between the two sub-sets. For the early generation, there are no specific patterns on the network architecture. On the other hand, for the late generation, we can see a specific similar pattern. Users can easily see when and how the evolution converged in terms of network architecture through this visualization. This visualization also helps the users understand the genetic algorithm and potentially provides insight into improving it.

The network architecture view also provides interactive analytics interfaces. In Figure 4, there are check boxes for each level. Users are able to filter the networks of interest by checking the check boxes. As the first two examples shown in Figure 6, the left image shows an example of filtering networks with convolution layers at the first and 11th levels. The other image shows an example of filtering networks matching to all the dominant layer types. The users can also select a specific network in the series of the networks and then the ancestry tree, links between the selected network and its parents appears as blue curves as shown in Figure 6 (Right).

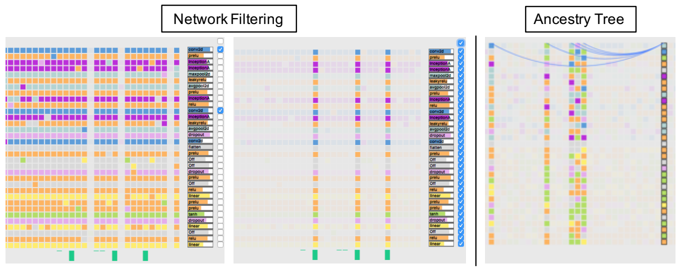

Fig. 6. Network Filtering and Ancestry Tree: Users can filter network of interest. Ancestry tree of the selected network provides lineage information.

VII. Conclusion

We have proposed a visualization system for evolutionary neural networks for deep learning and described what and how the three different views of the system visualize an evolutionary process. We have also demonstrated how interactive visualizations help users understand, examine, and improve a genetic algorithm. However, limitations still remain. It is still hard to find a causal relationship between fitness and hyper-parameters, even though comparing both values is available through the fitness-parameter view. Our system is not able to explicitly show why a genetic algorithm operates well or not in the evolutionary process. Also, the proposed visualization techniques have limitations. As future work, we will investigate ways to address the challenging tasks and improve the visualization techniques.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract number DE-AC05-00OR22725. This research is sponsored in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, ser. MLHPC '15. New York, NY, USA: ACM, 2015, pp. 4:1–4:5. [Online]. Available: http://doi.acm.org/10.1145/2834892.2834896

[2] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *ArXiv*, vol. abs/1712.06567, 2017.

[3] T. Salimans, J. Ho, X. Chen, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *ArXiv*, vol. abs/1703.03864, 2017.

[4] N. F. McPhee, M. M. Casale, M. Finzel, T. Helmuth, and L. Spector, "Visualizing genetic programming ancestries," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '16 Companion. New York, NY, USA: ACM, 2016, pp. 1419–1426. [Online]. Available: http://doi.acm.org/10.1145/2908961.2931741

[5] E. Hart and P. Ross, "Gavel - a new tool for genetic algorithm visualization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 335–348, Aug 2001.

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

[7] W. B. Shine and C. F. Eick, "Visualizing the evolution of genetic algorithm search processes," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, April 1997, pp. 367–372.

[8] H. Farooq, N. Zakaria, and M. T. Siddique, "An interactive visualization of genetic algorithm on 2-d graph," *Int. J. Softw. Sci. Comput. Intell.*, vol. 4, no. 1, pp. 34–54, Jan. 2012. [Online]. Available: http://dx.doi.org/10.4018/jssci.2012010102

[9] R. Miikkulainen, J. Z. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, "Evolving deep neural networks," *ArXiv*, vol. abs/1703.00548, 2017.

[10] H. Zhang, S. Kiranyaz, and M. Gabbouj, "Finding better topologies for deep convolutional neural networks by evolution," *ArXiv*, vol. abs/1809.03242, 2018.

[11] J. S. Yi, Y. a. Kang, and J. Stasko, "Toward a deeper understanding of the role of interaction in information visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1224–1231, Nov 2007.

[12] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw. Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, Nov. 1991. [Online]. Available: http://dx.doi.org/10.1002/spe.4380211102

[13] R. Wang, J. Clune, and K. O. Stanley, "Vine: An open source interactive data visualization tool for neuroevolution," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '18. New York, NY, USA: ACM, 2018, pp. 1562–1564. [Online]. Available: http://doi.acm.org/10.1145/3205651.3208236