Evolving Deep Networks Using HPC

Steven R. Young, Derek C. Rose, Travis Johnston, William T. Heller, Thomas P. Karnowski, Thomas E. Potok and Robert M. Patton
Oak Ridge National Laboratory
Oak Ridge
youngsr@ornl.gov

Gabriel Perdue
Fermi National Accelerator Laboratory
Batavia, IL
perdue@fnal.gov

Jonathan Miller
Universidad Técnica Federico Santa María, Avenida España
Valparaíso, Chile
Jonathan.Miller@usm.cl

ABSTRACT

While a large number of deep learning networks have been studied and published that produce outstanding results on natural image datasets, these datasets only make up a fraction of those to which deep learning can be applied. These datasets include text data, audio data, and arrays of sensors that have very different characteristics than natural images. As these "best" networks for natural images have been largely discovered through experimentation and cannot be proven optimal on some theoretical basis, there is no reason to believe that they are the optimal network for these drastically different datasets. Hyperparameter search is thus often a very important process when applying deep learning to a new problem. In this work we present an evolutionary approach to searching the possible space of network hyperparameters and construction that can scale to 18,000 nodes. This approach is applied to datasets of varying types and characteristics where we demonstrate the ability to rapidly find best hyperparameters in order to enable practitioners to quickly iterate between idea and result.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

high performance computing, deep learning, evolutionary algorithms, hyperparameter optimization

MLHPC'17, November 12–17, 2017, Denver, CO, USA

ACM ISBN 978-1-4503-5137-9/17/11…$15.00
https://doi.org/10.1145/3146347.3146355

ACM Reference format:
Steven R. Young, Derek C. Rose, Travis Johnston, William T. Heller, Thomas P. Karnowski, Thomas E. Potok and Robert M. Patton, Gabriel Perdue, and Jonathan Miller. 2017. Evolving Deep Networks Using HPC. In *Proceedings of MLHPC'17: Machine Learning in HPC Environments, Denver, CO, USA, November 12–17, 2017 (MLHPC'17),* 7 pages.
https://doi.org/10.1145/3146347.3146355

1 INTRODUCTION

There are several benchmark networks [12, 14, 25] that produce outstanding results on large natural image datasets such as ImageNet [21]. However, datasets of other modalities, like text and speech, as well as those built of images from other sources, such as microscopy [1, 16, 22] and neutrino detector data [3, 26], present unique modeling characteristics that can be very different than ImageNet. These characteristics have often required network hyperparameter choices distinct from previous work focused on natural imagery. Unfortunately, requesting domain scientists guess at the preferred deep network topology selection for their particular data can be daunting and inconvenient. A method to automatically discover the best networks for any problem domain is extremely valuable for efficiently applying deep learning models.

In this work, we present an evolutionary approach for the automatic discovery of performant networks for unique datasets and scale this approach to 18,000 nodes of Oak Ridge National Laboratory's Titan supercomputer. This framework is capable of not only evolving parameters like kernel size or the number of hidden units, but also of evolving the order, type, and number of layers within the network. The evolutionary algorithm used is asynchronous, allowing each node to always have a network to evaluate and maximizing the utilization of the machine. We apply this framework to a variety of unique datasets in order to demonstrate its ability to accelerate progress on new datasets.

2 RELATED WORK

Many techniques for addressing the problem of hyperparameter search in machine learning have been developed. Some practical guidelines that help form an intuition for manually constructing performant deep networks have been developed [5]. Gradient free approaches include grid search, random search [6, 8], coordinate descent [15], Bayesian methods [7, 23], and evolutionary algorithms [11, 18, 20, 24, 27, 28]. These methods rely on modeling from samples of hyperparameter configurations. Random and uncorrelated

search performs well in many cases due to the potential unimportance of a large number of hyperparameters and the ease of implementation for any algorithm.

Given the common design trends in deep learning (ResNet, Inception networks, U-Net), it is reasonable to assume that efficient network design can often be compartmentalized into modules or memes that appear as common network subgraphs in the topology. This motivates an evolutionary approach with strong random search tendencies where memetic subgraphs that perform well persist through design optimization. Related work in applying evolutionary algorithms to deep neural networks specifically has shown recent success in configuring topologies [20]. [27] built networks focusing on learning appropriate connections for convolution layers. [11] utilizes an evolutionary algorithm to select pathways to form a deep network from a fixed set of modules for classification and reinforcement learning tasks. [20] shares many similarities to the work presented here, where an evolutionary process mutates network graph configurations. In their work, weight inheritance is leveraged to speed up evaluation and stalling is avoided with tournament selection among 250 parallel workers. [18] modified a neural network topology evolution method, NEAT [24], to build deep networks by constraining the original genome encoding to layers. This work further built modularly oriented deep designs by co-adapting network sub-modules and blueprints for layout (and common reuse) of discovered modules. Our work and these related works do not use evolution to learn the underlying network weight parameters as well, opting instead to leave this to traditional gradient descent based methods.

"Hypergradient" based methods [4, 17] have been developed using principles similar to backpropagation for network parameters to adjust network weights. This style of optimization provides an interesting opportunity to challenge design intuitions behind learning rate schedules in optimizers and fundamental architecture choices like weight sharing. Unfortunately, this technique can be limited by gradient descent pitfalls, memory for the update trace, and unclear paths for optimizing discrete and ordinal parameters.

3 METHOD

Our approach to network hyperparameter search is to utilize an evolutionary algorithm. The goal of our algorithm is to identify optimal hyperparameter sets in a few hours on a petascale machine, thus we made design choices intended to leverage these resources. We needed to utilize an asynchronous evolution strategy in order to increase utilization on the machine. The computational resources available on the target machines provide ample opportunity to search the hyperparameter space beyond a simple optimization of a subset of hyperparameters within a fixed topology. Our gene structure of the individuals needed to be able to exploit the search opportunity.

3.1 Asynchronous Evolution

In related work, [28], a synchronous evolutionary algorithm was utilized. Although the variety of networks that could be created was rather limited, vast disparities in fitness evaluation times occurred and resulted in poor resource utilization. In our work, an asynchronous evolutionary algorithm is utilized such that each

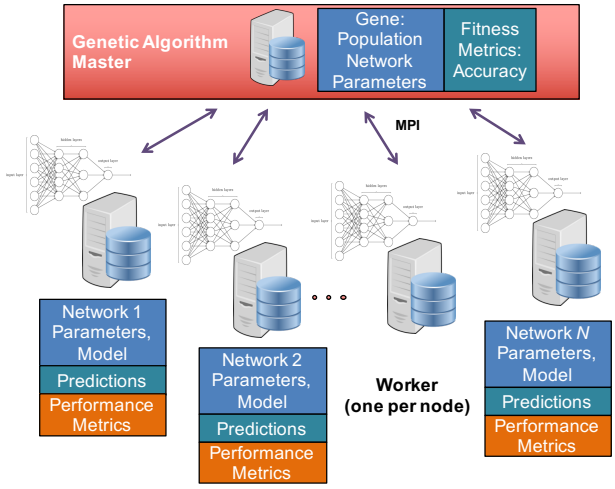

Figure 1: The evolutionary algorithm uses a master/worker system where the master hosts the selection, crossover, and mutation processes, while the workers evaluate the fitness of the individuals.

node is continuously evaluating network fitness rather than waiting for all of the other nodes to finish evaluation of their respective individuals. One node is used to host the selection, mutation, and crossover processes, while the remaining nodes evaluate the fitness of an individual by training and evaluating the network using Caffe [13]. This master/worker configuration is shown in Figure 1. The evolutionary algorithm used in this work is identical to that presented by [28], except that selection is performed whenever n of the N members of the population have been evaluated, where $n = N/3$. Assuming the evaluation time for any single individual is always longer than the time to perform selection, mutation, and crossover for the entire population of evaluated individuals, $n = N/2$ could be used, but in order to account for the case where this may not be true, we do not set the bounds that tight in order to allow for some individuals that may be evaluated extremely quickly. This ensures each worker always has a network to evaluate. This is demonstrated in Figure 2.

Additionally, this asynchronous nature provides an opportunity to exploit and discover networks that train faster since faster training networks may evolve unimpeded by the slowest networks. A faster iteration allows quick training networks to constitute a larger portion of the evolutionary process, thus biasing the solution towards networks which reach high accuracies earlier. Otherwise, the evolution process is the same as is used in [28], where above average selection is used and the best network persists in the population for selection.

3.2 Genetic Encoding

Defining a robust gene configuration for network topologies is challenging when attempting to efficiently utilize shared compute resources. It is preferable that the number of network topologies that fail to compute be minimized while retaining a high variability of generated networks. Failures are caused by improper network

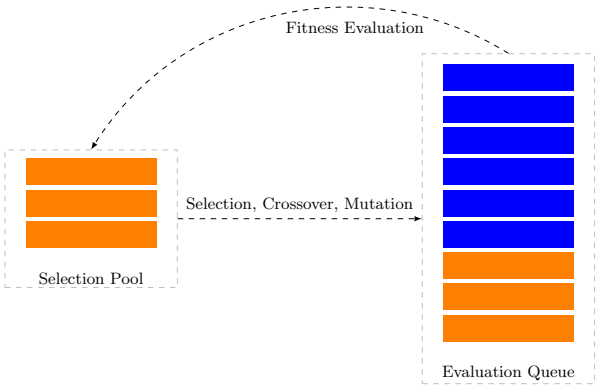

Figure 2: This figure demonstrates the asynchronous evolutionary algorithm used in this work. This approach ensures there are always individuals available in the queue to be evaluated.

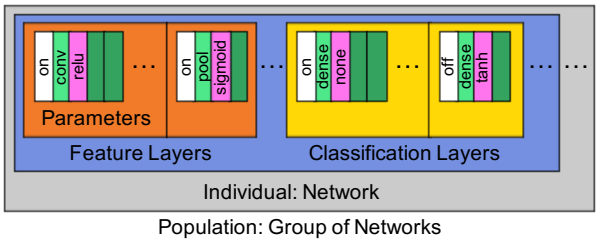

Figure 3: Hyperparameter organization for population. Each layer has 3 genes which encode layer on/off, layer type, and chained activation type.

connections or inputs sizes, such as choosing a convolution layer with a filter size of 7×7 on input patches of 5×5. Another such failure would occur if a fully connected layer was fed into a 2D convolution layer. To prevent the latter and reduce the hyperparameter search space, we have explicitly structured the network gene into two parts for the task of image or pixel-wise classification.

Figure 3 shows the layout of the genetic map for the EA utilized. The first part of the gene describes what we deem as "feature" layers. These include convolution, pooling, normalization, etc. layers which work on 2D maps derived from the original input images. Fully connected layers explicitly compose the second part of the gene and lead to eventual classification targets. For other data modalities or fully convolutional networks, the genetic encoding can be logically extended.

During implementation we avoided creating multiple random generators with restrictions that would need to be individually mapped to gene positions. We instead employed one "DNA" generator which sampled from signed four byte integers extending to the maximum integer value. During mapping, a small set of encoding functions converted the gene from integers given the requirements of each layer hyperparameter. Encoding functions converted integers to floating point ranges, $x \in [a, b] \in \mathbb{R}$ for hyperparameter x. If integers are required, $x = \lfloor x \rfloor$. For ordinal hyperparameters (e.g. pooling type and weight initialization), integers were mapped to the range of enumerated types.

3.2.1 Network Gene Layout. For each layer, a type enumeration and layer on/off bit begins the gene. Each layer also has a marker for an activation function which is chained to the layer. The encoded hyperparameters which follow have a fixed count at a ceiling which allows the layer type with the most hyperparameters (in our experiments, 11 for convolutional layers) to be expressed. Each layer type uses the encoding in these variables in a distinct way. The unbound encodings are inactive and ignored when the gene is converted to a final network description. This results in two "meta" hyperparameters for our gene layout: 1) a fixed number of possible layers, and 2) the probability of turning any single layer on.

3.2.2 Connection Concerns. While a two part mapping avoids configuration failures from interlacing convolutional and dense layers, layers of incompatible output and expected input dimensions would still be built through a completely random selection. To avoid this, during network construction we keep track of the current layer's input dimensions and appropriately limit the selected hyperparameters (e.g. pool kernel size) to fall within the usable range. The next layer's input dimensions are computed after the selection of hyperparameters that modify the shape of the data flowing through the network. With this technique a progressive build occurs such that a change in an early layer gene could cascade to force a different kernel size configuration in later layer genes. Under this design, crossover effectively causes an inheritance of layer types and parameters other than kernel sizes. The range encoding with dimensionality constraints smoothly limits inherited kernel size traits to those which fit within a new topology.

4 DATASETS

We evaluate our method on a variety of datasets described in the following subsections. Each dataset varies in the task being performed, the characteristics of the images, the number of examples available for training, and the dimensionality of the images. They were chosen such that our method could be evaluated across a wide range within each of these dataset properties.

4.1 Neutrino Detector Vertex Reconstruction

MINERvA (Main Injector Experiment for v-A) [19] is a neutrino scattering experiment at Fermi National Accelerator Laboratory. The detector is exposed to the NuMI (Neutrinos at the Main Injector) neutrino beam. Energy values collected from the detector may be mapped to pixel values in three images, with each image collected at a different angle relative to the detector and subsequently analyzed. The location of the event vertex is an important to identify for analysis. Deep learning has been shown to be a viable alternative to traditional methods for this problem[26]. For this work, we look only at simulation images generated for a single view out of the three available and the task is to correctly identify which of 11 regions the vertex of the event is located in. There are 800,000 images available for training with each image of size 127×50.

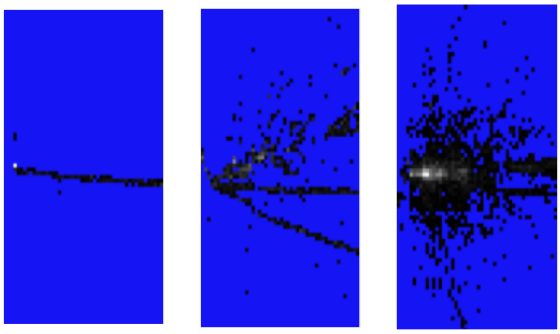

Figure 4: Three examples of single view data from the MINERvA detector. The examples demonstrate the range of complexity in the events that occur in the detector.

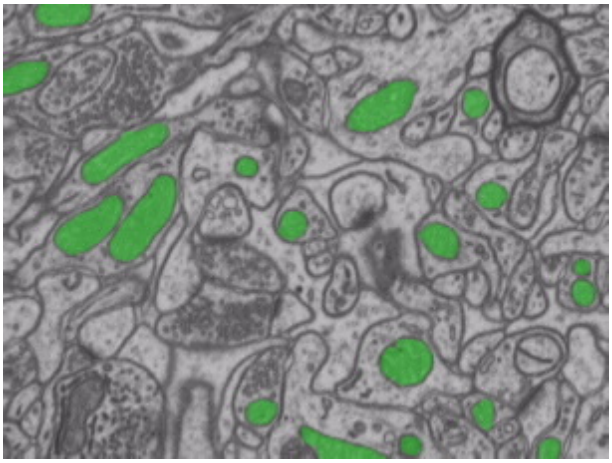

Figure 5: Mitochondria segmentation data. The green areas are the regions classified as mitochondria by a deep learning network.

Figure 4 shows three example images from this dataset that capture the various characteristics of events that occur within the detector.

4.2 Pixel-wise Segmentation of Mitochondria

We tested our proposed method on a network designed to segment mitochondria in electron microscopy images. Labeled data consists of 3D image stacks ($1024 \times 768 \times 165$) and is publicly available from EPFL [2]. We split the training stack of images vertically into a training and validation set, sampling $300,000$ 62×62 2D patches from each section. We performed hyperparameter selection and evaluation on the validation patches. Figure 5 shows an example image from the testing set. 3D EM data presents challenges in its scale, resolution, and unbalanced nature, where most of the pixels composing the image belong to background.

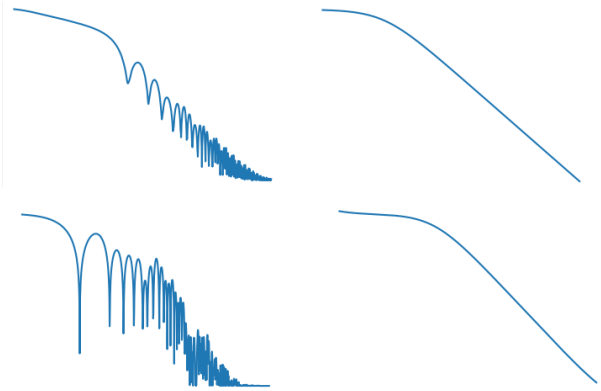

Figure 6: Examples plots of small angle scattering plots corresponding to different physical models: (top left) a cylinder having a core and shell with different densities, (bottom left) spherical sld, (top right) a Gaussian polymer chain having a distribution of lengths, (bottom right) correlation length

4.3 Neutron Scattering Model Selection

The final test of our proposed method was on a network designed to categorize the type of physical model that gives rise to small angle neutron scattering data. Classifying such data can be challenging for novice users of the technique due to the similarity in appearance. Novices generally consult with an expert by first showing them a plot of the data. Simulated data was generated using an extensive set of functions for modeling small angle scattering data and SasModels, which is a part of the SasView software[9], and corresponds to 9 different classes of models with $1,000$ examples of each class available for training. Each class of model corresponds to a different type of shape such as a cylinder, a sphere, or a polymer chain displaying Gaussian statistics. The data takes the form of scattering intensities measured as a function of the scattering vector magnitude. Inspired by work in speech recognition where sound recordings are plotted as spectrograms [10], this data can then be plotted as a line plot (200×600pixels) and supplied as input to a convolutional neural network. Some example data for this problem are shown in Figure 6 and it can be seen that very different structures can give rise to data that look very similar. Classifying these plots is particularly difficult for anyone who is not a subject matter expert.

5 RESULTS

The evolutionary algorithm improved results over our baseline on each of the target datasets. We were able to achieve this improvement in a matter of hours, which will allow researchers to focus on how best to frame their problem for deep networks, rather than focusing on what hyperparameters work best.

5.1 Neutrino Detector Vertex Reconstruction

A simple network configuration was used to seed our method and originally achieved a vertex classification accuracy of 77.88%, which we were able to evolve to an 82.11% accuracy. In 26 generations of evolution, we evaluated approximately $500,000$ convolutional

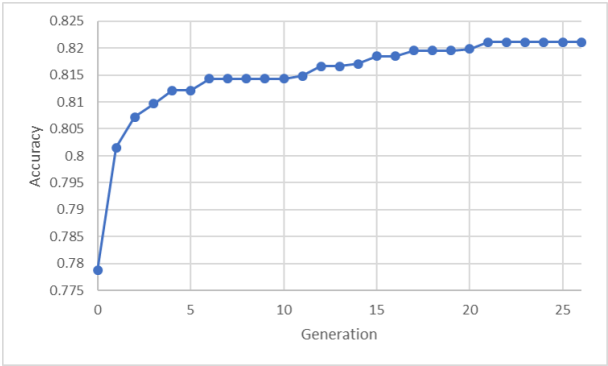

Figure 7: Best accuracy (fitness) versus iteration for the neutrino vertex classification dataset. Generation 0 is the seed network. Generations 1-3 are the initial randomly mutated population. The remaining generations are the products of selection, mutation, and crossover.

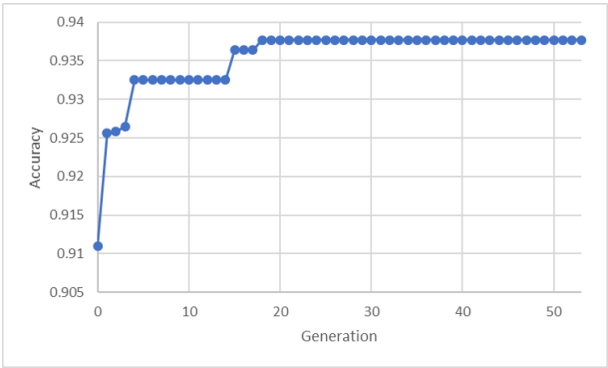

Figure 8: Best accuracy (fitness) versus iteration for the mitochondria dataset. Generation 0 is the seed network. Generations 1-3 are the initial randomly mutated population. The remaining generations are the products of selection, mutation, and crossover.

networks on 18,000 of Titan's nodes in 24 hours, with the best network being found in 19 hours. Figure 7 shows the accuracy of the best network vs the generation count. This dataset demonstrates that our method is capable of utilizing a human defined seed and evolving that seed into a better performing network.

5.2 Pixel-wise Segmentation of Mitochondria

A human designed network configuration was used to seed our method and originally achieved a pixel-wise binary accuracy of 91.10%. Through evolution our method discovered a network capable of 93.77% accuracy, representing a 30% reduction in error vs. a human expert defined network. Our evolutionary algorithm ran for 36 generations and evaluated over 900,000 convolutional networks on 18,000 of Titan's nodes for 24 consecutive hours, with the best network being found in 11 hours. Figure 8 shows the accuracy of the best network vs the generation count. This figure demonstrates that the evolutionary algorithm provides an improvement over random search on this problem as generation 0 represents the seed network, generations 1-3 represent the initially randomly mutated seeds, and the remaining generations are the product of selection, crossover, and mutation. The randomly generated networks only provide a small increase in performance in generations 2 and 3, and the jump in performance from generation 3 to generation 4 demonstrates the impact of evolution.

5.3 Neutron Scattering Model Selection

Our evolutionary algorithm evaluated nearly 700,000 networks on 18,000 nodes of Titan in 24 hours for the neutron scattering data, and found a best network that achieved 76.07% accuracy, which is an improvement over the initial seed's 68.71% accuracy. Unlike the other two systems studied, this best network was found simply through mutations of the initial seed without performing any crossover. This rapid optimization likely resulted from the extremely small training set that consisted of only 9,000 training examples, which demonstrates that the quality of the dataset can limit the ability of the evolutionary process to improve accuracy. Another interesting result observed on our testing on this dataset also arises from the small number of training examples. The best network has a dropout layer very close to the data layer in the network as shown in Figure 9, which essentially functions a form of data augmentation, because activations located very close to the data are dropped. Such behavior is atypical of most networks published using larger datasets where dropout is typically only used in later, fully connected layers.

6 CONCLUSION AND FUTURE WORK

We have presented a framework for evolving deep learning networks at an extreme scale that is well suited for rapidly discovering performant networks for new, unique datasets. We demonstrated this framework on a sampling of datasets that are unique from the natural image datasets typically seen in literature.

Future work will seek to further unconstrain the network topology such that more complicated graph structures can be evolved. The current framework assumes that each layer is stacked upon the previous layer. However, there are several benchmark networks that use more complex branching structures [25]. By incorporating related work that is able to evolve graph structures, in our case with a layer being a node in the graph, we hope to evolve complex multi-path networks and identify other useful subgraphs. As an intermediate step toward more complex networks, we plan to extend the genetic encoding to add a bridging layer type that allows skip connections, as these have proven especially useful for Fully Convolutional Networks and ResNet. We additionally look to expand our gene for each network to include hyperparameters related to the network optimizer, such as learning rate and policy, momentum, backpropagation type, etc. These hyperparameters do increase the search space significantly but could lead to networks which are otherwise missed given the bias inherent in fixed choices for training parameters.

These experiments have resulted in a rich meta-dataset consisting of a variety of network topologies and their corresponding performance on a variety of problems. Analysis of this unique dataset

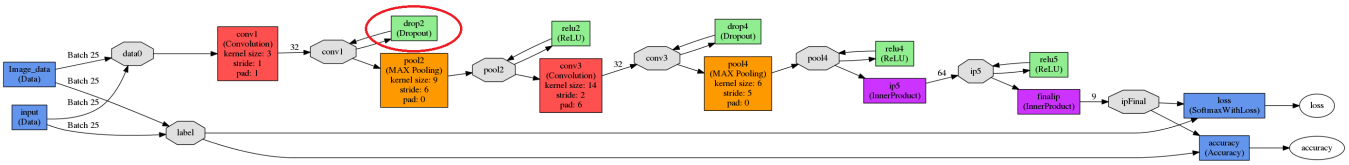

Figure 9: Best network for the neutron scattering dataset. Note the dropout layer early in the network that is essentially acting as a data augmentation unit due to the small size of the training set.

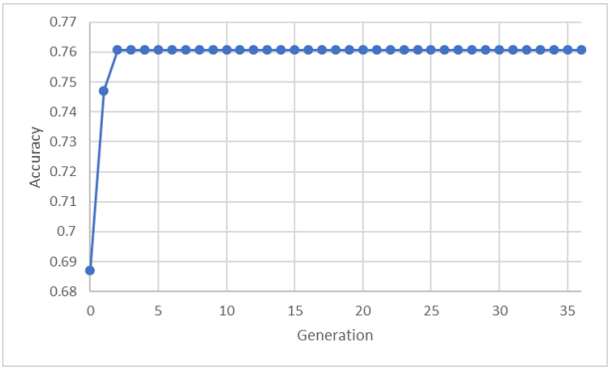

Figure 10: Best accuracy (fitness) versus iteration for the neutron scattering dataset. Generation 0 is the seed network. Generations 1-3 are the initial randomly mutated population. The remaining generations are the products of selection, mutation, and crossover.

gives the opportunity to begin to explore the complex relationship between hyperparameters, datasets, and performance.

7 ACKNOWLEDGEMENTS

Research sponsored 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.

We would like to thank the MINERvA collaboration for the use of their simulated data and for many useful and stimulating conversations.

MINERvA is supported using the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, HEP User Facility. Fermilab is managed by Fermi Research Alliance, LLC (FRA), acting under Contract No. DE-AC02-07CH11359, which included the MINERvA construction project. Construction support also was granted by the United States National Science Foundation under Award PHY-0619727 and by the University of Rochester.

Support for participating scientists was provided by NSF and DOE (USA) by CAPES and CNPq (Brazil), by CoNaCyT (Mexico), by Proyecto Basal FB 0821, CONICYT PIA ACT1413, Fondecyt 3170845 and 11130133 (Chile), by CONCYTEC, DGI-PUCP and IDI/IGI-UNI (Peru), by Latin American Center for Physics (CLAF) and by RAS and the Russian Ministry of Education and Science (Russia).

This work benefited from the use of the SasView application, originally developed under NSF award DMR-0520547. SasView contains code developed with funding from the European Union's Horizon 2020 research and innovation programme under the SINE2020 project, grant agreement No. 654000.

REFERENCES

[1] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. 2016. Deep learning for computational biology. *Molecular Systems Biology* 12, 7 (July 2016), 878.

[2] Carlos Becker Pascal Fua Aurelien Lucchi, Yunpeng Li. [n. d.]. École polytechnique fédérale de Lausanne: Electron Microscopy Dataset. ([n. d.]). http://cvlab.epfl.ch/data/em

[3] A Aurisano, A Radovic, D Rocco, A Himmel, MD Messier, E Niner, G Pawloski, F Psihas, A Sousa, and P Vahle. 2016. A convolutional neural network neutrino event classifier. *Journal of Instrumentation* 11, 09 (2016), P09001.

[4] Atilim Gunes Baydin and Barak A. Pearlmutter. 2014. Automatic differentiation of algorithms for machine learning. *arXiv preprint arXiv:1404.7456* (2014).

[5] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer, 437–478.

[6] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.

[7] James Bergstra, Brent Komer, Chris Eliasmith, and David Warde-Farley. 2014. Preliminary evaluation of hyperopt algorithms on HPOLib. In *ICML workshop on AutoML*.

[8] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*. 2546–2554.

[9] P Butler, G Alina, RC Hernandez, M Doucet, A Jackson, P Kienzle, S Kline, and J Zhou. 2013. SASView for Small Angle Scattering Analysis. (2013).

[10] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. 2013. Recent advances in deep learning for speech research at Microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 8604–8608.

[11] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734* (2017).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[13] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 675–678.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[15] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. 2007. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*. ACM, 473–480.

[16] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken, and Clara I. Sánchez. 2017. A survey on deep learning in medical image analysis. *arXiv preprint arXiv:1702.05747* (2017).

[17] Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*. 2113–2122.
[18] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. 2017. Evolving Deep Neural Networks. *arXiv preprint arXiv:1703.00548* (2017).
[19] GN Perdue, L Bagby, B Baldin, C Gingu, J Olsen, P Rubinov, EC Schulte, R Bradford, WK Brooks, DAM Caicedo, et al. 2012. The MINERνA data acquisition system and infrastructure. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 694 (2012), 179–192.
[20] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc Le, and Alex Kurakin. 2017. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041* (2017).
[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.
[22] Dinggang Shen, Guorong Wu, and Heung-Il Suk. 2017. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering* 0 (2017).
[23] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.
[24] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.
[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.
[26] Adam M Terwilliger, Gabriel N Perdue, David Isele, Robert M Patton, and Steven R Young. 2017. Vertex reconstruction of neutrino interactions using deep learning. In *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2275–2281.
[27] Lingxi Xie and Alan Yuille. 2017. Genetic CNN. *arXiv preprint arXiv:1703.01513* (2017).
[28] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. 2015. Optimizing Deep Learning Hyper-parameters Through an Evolutionary Algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*. ACM, 4.