Deep Squeeze Evolution of Convolutional Neural Network Hyperparameters for Radiographic Detection of CoronaVirus (COVID-19)

David James Gabriel
Department of Computer Science and Engineering
University of Nevada Reno
Reno, Nevada
contact@david-gabriel.com

Sushil J Louis
Department of Computer Science and Engineering
University of Nevada Reno
Reno, Nevada
sushil@cse.unr.edu

***Abstract*—CoronaVirus Disease 2019 (COVID-19) has impacted global health and economy in a paradigm shifting global pandemic. Globally, there have been 68,165,877 confirmed cases of COVID-19, including 1,557,385 deaths, reported to the World Health Organization (WHO) as of December 10, 2020 [1]. Critical to containment of COVID-19 are diagnostic testing methods. Large scale testing is crucial to early detection, however supply chain shortages of microbiological agents have created pandemic response challenges [2]. Radiographic Imaging diagnostic methods of COVID-19 have shown some efficacy outside of clinical trials using convolutional neural networks (CNNs) [3].**

We compare methods for hyper parameter search on CNNs. Deep Evolution (DEvol) [4] surpasses Hill Climber performance on MNIST handwriting classification dataset [5]. We propose modified DEvol (mDEvol) and adapt mDEvol to SqueezeNet fire layers [10]. SqueezeNet fire layers provide a reduction of the number of hyperparameters for equivalent model accuracy, by using an expansion operator [10]. We apply Evolving Deep Squeeze for hyperparameter optimization of CNN's for COVID-19 radiographic image diagnostics.

In a pandemic response, total time to deployment of a diagnostic tool is critical to coordinating mass testing [2]. Because the cost of exhaustively searching the hyperparameter space of a CNN is on the order of thousands of years for the simplest cases of MNIST, acceleration methods are reviewed. State of the art parallelization performance is demonstrated on a distributed Apache Spark computing cluster using NVIDIA Tesla P100 Graphics Processing Unit (GPU) acceleration. Superlinear speedup is realized through use of Spark Resilient Distributed Dataframe (RDD) parallelization [6]. Speedup factors up to 15.15x over 8 core multi threaded Central Processing Unit (CPU) execution using two GPUs each on six computational nodes. We are able to show superior speedup on computational nodes with fewer GPUs per node and more nodes total. This is because DEvol and its variants are primarily GPU memory bandwidth limited for the cases evaluated.

The mDEvol is applied to a new genome configuration for the COVIDX dataset of COVID-19 images. The mDEvol is combined with SqueezeNet for Deep Squeeze Evolution (DSE). DSE is used as a hyperparameter optimizer to produce a CNN diagnostic tool.

Keywords—CoronaVirus, COVID-19, Evolutionary Computing, Convolutional Neural Networks, Artificial Intelligence, Machine Learning, Deep Learning, Classification, Image Recognition, Automated Diagnosis, Apache Spark,Parallel Computing

I Introduction

COVID-19 has killed over 1.57 million people globally and is still an active global pandemic as of December 2020 according to the World Health Organization (WHO) [1]. The shortage of microbiological agents in the global supply chain, which has led to diagnostic testing shortages, complicates response efforts. Because early stage pandemic response relies so heavily on contract tracing, rapid, scalable testing is critical to pandemic response [2]. Non-contact testing methods, particularly those which do not consume any quantity of microbiological agents in the evaluation of a test have particular advantages when responding to a global pandemic. First, they reduce strain on protective personal equipment supplies and possible transmission vectors in the testing process by eliminating direct contact during sample collection. Second, they do not consume microbiological agents, further contributing to shortfalls during peak demand for these critical materials. Thirdly, such methods may return a diagnosis more rapidly in some cases, and reduce demand on microbiological agent processing laboratories. Techniques are reviewed specifically for radiographic diagnosis of COVID-19 and analysis of the pre-clinical efficacy with aims to prevent further harm from this and future global pandemics.

Convolutional Neural Networks (CNNs) are used in image recognition tasks to determine class membership. Through a learning process, the CNN is able to identify images as belonging to a set, such as handwritten numbers. The same learning process that allows a CNN to learn to recognize handwriting can be used to train a CNN for COVID-19 detection. A system designed around CNN based diagnosis, if successful, would be able to perform diagnosis without the supply chain demands, and with

rapid throughput of non-contact testing. Also, this system could be deployed to hospitals around the world via software using equipment they already have, further reducing infrastructure related pathogenic transmission.

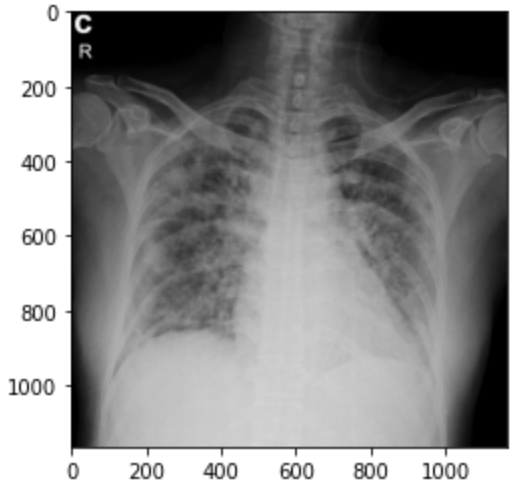

Fig. 1. Chest X-ray (CXR) image from COVIDX dataset.

Recent developments in cloud computing and Artificial Intelligence (AI) have produced development platforms with advanced learning systems, such as Keras and Apache Spark [7,6]. These systems allow very complicated models to be developed with relatively little code, as long as there is well labelled reference data from which the models may learn. However, the effects of model hyperparameters, such as the type and number of convolutions in each layer, and the structure of layers in the CNN are poorly understood, especially for auto-diagnostic applications [9]. These hyperparameters have a large number of permutations for even simple classifiers in the image domain. Additionally, evaluating a set of hyperparameters is computationally expensive, with the dataset shown to the model in multiple epochs, and model learning adjustments calculated throughout. *We use a Genetic Algorithm methodology, which encodes the hyperparameter space into a genetic representation, and searches through the space using a population of competing individual models.*

In order to further expedite deployment time of Genetic Algorithm developed radiographic diagnostic systems, we examine the embarrassingly parallel nature of genetical search, and exploit generational parallelism of fitness evaluation overlaying upon data parallelism at the individual level. Speedup of 15.15x for a 3 genome training scenario is achieved by comparing Microsoft Azure Dsv3 node to Azure NC12s_v3 with Apache Spark Resilient Distributed Dataframe (RDD) execution over an accelerated compute cluster. We examine autoscaling of parallel resources and demonstrate superlinear speedup relative to the number of cores. This form of generational parallelism has potential advantages in both training and batch processing of diagnostic tests.

Chest X-Ray (CXR) images of COVID-19 patients, anonymized and with relevant metadata is provided in [11]. This dataset contains images for training CNNs for COVID-19 diagnosis. In order for CNN diagnostics to provide viable improvements in pandemic response, it is necessary to have quality sources of data, in addition to validation and transmission capability relevant to local regulations.

In light of COVID-19, rapid deployment of CNN modelling techniques for pandemic diagnostic response via CXR images is considered. Special attention is considered to the practical requirements of pandemic response, such as rapid network training, diagnostic testing throughput, statistical performance metrics, lostical deployment issues, and capability to operate on standard computing hardware or on cloud based service platforms.

II Prior Work

Deep Evolution (DEvol) has been shown to be able to evolve MNIST handwriting recognition successfully [4]. DEvol is a CNN hyperparameter optimization algorithm which encodes CNN hyperparameters into a genome, and uses the properties of mutation and recombination to evolve individuals with increasing fitness for image classification or other purposes. DEvol is built on top of Keras. DEvol encodes Keras sequential 2 dimensional CNN's into a fixed number of main layers, which may be of the type convolutional or deep. Additionally, max pooling layers and dropout modifiers are optionally encoded into one of the main layers, as are activation functions and other hyperparameters. The final layer is always a dense layer of type softmax, because this is necessary to normalize output. An optimizer is also encoded.[4]

In [12], the authors review characteristics of evolutionary methods including DEvol, and propose their evolutionary system for Internet of Things sensor data modelling of human activity.

CoroNet is shown in [13] as a CNN diagnostic model for COVID-19 detection in CXR images. CoroNet uses the COVIDX dataset with random undersampling and transfer learning methodology. In a 4-class comparison, CoroNet is shown to be superior to COVID-Net[14] in both greater accuracy and fewer numbers of parameters.

SqueezeNet has been proposed as a COVID-19 diagnostic tool in [9], where the authors demonstrate Bayesian Hyperparameter Optimization of SqueezeNet and Data Augmentation methods. SqueezeNet uses fire layers to reduce the size of hyper parameter encodings necessary to construct a CNN of a given complexity, by utilizing an expansion and squeeze operator to generate larger network complexity relative to hyperparameter encoding space[10]. This is beneficial in reducing deployment time of a diagnostic solution. Additionally it reduces the file size of the final model, which has other advantages in light of pandemic response for testing throughput and deployment.

In [15], the authors review data center focused methods for CNN training and inference, in relation to the challenges of running Facebook inc.. The authors show the benefits of GPU accelerated training with data parallelism, and demonstrate bottlenecks of operation in communication between parallel cores. The authors show advanced data center distribution techniques similar to those implemented in the open source Apache Spark. [6,15] It is shown that the bottleneck of execution in CNN parallel training is bandwidth between cores[15], which is either network bandwidth or GPU memory bandwidth in different execution contexts.

III METHODOLOGY

III.A Evolutionary Methods

DEvol uses single point crossover, and a mixture of 5% elitist selection and 95% single child fitness proportional selection to produce successive generations. DEvol uses mutation warm up for increasing mutation activation after 12 generations. No fitness scaling is used. The genome encoding specifies a maximum number of layers of convolutional and dense types, as well as a number of nodes in the dense layer and a number of filters in the convolutional layers. A softmax layer is always the last layer to ensure correct unary sum output. Additionally dropout and max pooling layers are encoded as modifiers to the convolutional layers. There are 4 possible optimizer functions. [4] This creates the following size of search space for a DEvol encoding of CNN hyperparameters:

$$S = 4 * (26 * N_{filters})^{L_{conv}} * (24 * N_{nodes})^{L_{dense}-1} \quad (1.)$$

Eqn. 1. S is the size of the hyperparameter search space, L_{conv} is the number of convolutional layers and L_{dense} is the number of dense layers. $N_{filters}$ is the maximum number of filters in a convolution and N_{nodes} is the maximum number of nodes in a dense layer.

DEvol is tested in its ability to perform hyperparameter optimization compared to a gradient ascent optimizer and we propose a modified form of DEvol (mDEvol). We compare multiple runs of each algorithm, tracking the average cumulative maximum fitness and average mean fitness vs. number of evaluations (evals). mDEvol adds additional exploration pressure, which is compared without modification to the elitism percentage of 0.05% in all cases on MNIST. These additional exploration dimensions, while adding complexity to the genetic parameters, allow for further turning of selection pressure vs exploration. This is consistent with techniques demonstrated in [16], where the author shows an alternative genetic algorithm design technique using more conservative selection and disruptive crossover and mutation and demonstrates increased performance on a wide range of problems. In mDEvol, the addition of uniform crossover and population relative mutation warm up are done with the end goal of producing an algorithm using these increased convergence pressure and increased disruption genetic techniques. We demonstrate a characteristic change in evolutionary patterns between DEvol and mDEvol.

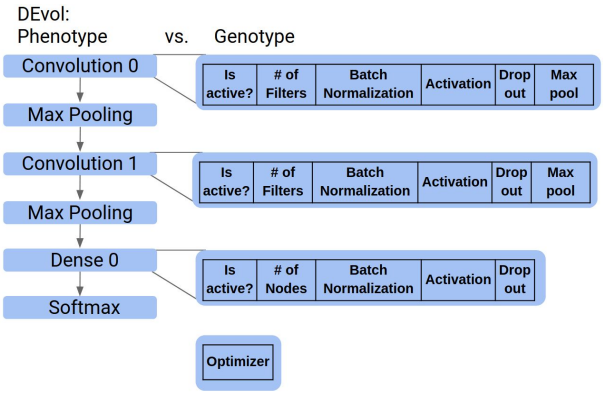

Fig. 2. Phenotype (left) vs Genotype (right). This is for a CNN with $L_{conv} = 2$ and $L_{dense} = 2$.

The parameters of genetic search themselves bear some similarity to CNN hyperparameters in that they must be optimized, incur great computational expense, and are not well understood in many cases. A tradeoff exists between the number of genetic parameters and CNN hyperparameters. Both must be optimized, so we must be careful not to simply shift the problem into a different domain. Using a fewer number of genetic algorithm parameters, it is possible to parallel search a very large number of CNN hyperparameters. Therefore, in the context of pandemic response, genetic optimization of CNN hyperparameters benefit in both speed of and scalability of implementation and distribution.

We propose Deep Squeeze Evolution (DSE), which combines SqueezeNet with Deep Evolution and is applied to search the hyperparameter space for optimal SqueezeNet parameters. DSE is designed for parallel search of challenging and computationally expensive hyperparameter optimization problems. DSE attempts to improve upon Bayesian hyperparameter tuning principles demonstrated in [9] with advanced parallel genetic search. Within the context of pandemic response, the advantages of minimal speed of deployment and best pre-clinical accuracy and precision are considered. The genetic aspects of DSE allow it to evaluate more of the search space than equivalent computationally expensive Bayesian methods. This may be exploited for our purposes of rapid diagnostic deployment. Additionally, the ratio of performance to genome size in improved with selection of SqueezeNet topology for DSE.

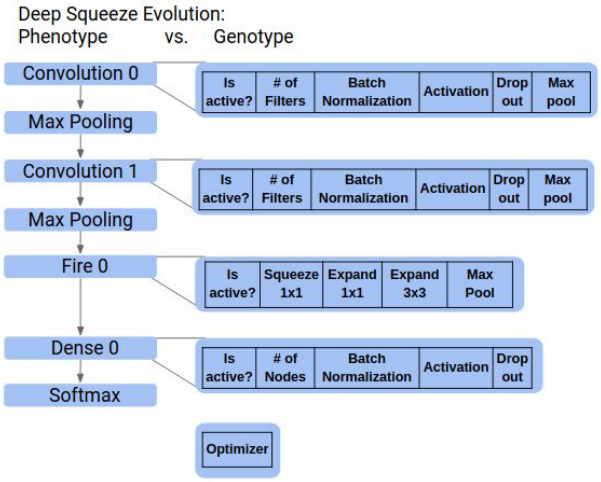

Fig. 3. Phenotype (left) vs Genotype (right). For a CNN with L_{conv}=2 and L_{dense} = 2 and L_{fire}= 1, where L_{fire} is the number of fire layers. In the simplest implementation all non binary fire parameters = # of convolutional filters

DSE adds another layer type to DEvol which is the fire layer. The fire layer uses squeeze and expand nodal structure to achieve comparable performance to CNNs with fifty times the number of hyperparameters encoded. [10] This has the effect of improving the ratio of search time to performance, which equates to more efficient pandemic response protocol.

$$S = 4*(26*N_{filters})^{L_{conv}}*(24*N_{nodes})^{L_{dense}-1}\ldots$$
$$*(4*S_{1x1}*E_{1x1}*E_{3x3})^{L_{fire}} \quad (2.)$$

Eqn. 2. We see the addition of L_{fire} , the number of fire layers. S_{1x1} , E_{1x1} , E_{3x3} are the maximum number of 1x1 and 3x3 squeeze and expand modules in the fire layer.

III.B Parallelization Methods

Alacrity of deployment being critical to successful pandemic response, we evaluate state of the art acceleration methods. The shortest evaluation time recorded using non GPU accelerated hardware was on a Dsv3 with eight (8) cores averaging times for a three individual evaluation. This was 6.46 minutes per individual trained on average (σ =0.040). The same individuals were randomly chosen and used in every timing evaluation. The number of three individuals was chosen, because it is more challenging a benchmark than larger numbers of individuals per trial in which to show speedup. GPU parallelism must overcome the costs of data transfer and benefits from larger parallel job sizes with more individuals as such. By using three individuals we are able to ensure speedup in all cases of populations larger than this all else being equal. Table 1 shows the hardware configurations used.

Microsoft Azure Hardware Specifications

Quantity of Type	6	1	1
Type	NC12s_v3	NC24s_v3	D8s v3
vCPU	12	24	8
Memory: GiB	224	448	32.00 GiB
GPU	2	4	0
GPU memory: GiB (each)	32	64	-
Total Cores	3852	2584	8
Cost / Instance (DBU)	3	6	1.5
Average Train Time (minutes/3 individuals)	1.28	19.39	2.58
Train Time σ =	0.027	0.121	0.132

Table. 1.Harware specification table for Apache Spark Databricks cluster configured in Microsoft Azure cloud with NVIDIA Tesla P100 GPUs in variable configurations. Cost in Databricks Units (DBU), price metric linear in time [7].

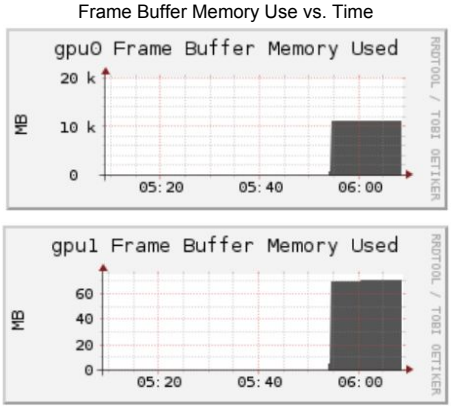

Fig. 4. Ganglia metrics were used to track GPU execution during MNIST search. GPU memory bandwidth is the limiting factor, seen here resulting in flat Frame Buffer usage across GPUs during execution on 6 NC12 nodes. Metrics via [9]

III.C Case Injection of Initial Conditions

SqueezeNet has been shown as a viable CNN topology for COVID-19 detection [3]. Case injection has been shown to improve performance over random initial

conditions when solving problems with a defined similarity. In the case of a CXR diagnostic, when a new disease arises, a history of CNN CXR diagnostics could serve as a case base from which to perform case injection for increased genetic search performance. This again is considered within the context of a pandemic response, as a potential reduction in deployment time for such a system. In the context of a pandemic response, using case injection which has learned from previous diagnostic information may accelerate training times when doing so is critical.

A well developed case base may accelerate CNN hyperparameter tuning. The layer structure used in [3] is the most applicable candidate for case injection, and as such is used in DSE trials as an initial condition for 5% of the population. The DSE encoding is slightly different. The closest DSE encoding available is one convolutional layer, eight fire layers and two dense layers. Because DSE must specify an upper bound to each number of layers. SqueezeNet architecture typically relies on fire layers as the main layer type, as seen in [3]. We therefore add an additionally two fire layers to the upper bound. Choice of additional layers must be done sparingly, as the increase in search space size scales rapidly. Fire layers provide increased utility in an efficient exchange for additional complexity [10].

IV Results and Discussion

mDEvol is shown to have increased exploration from uniform crossover. mDEvol also uses mutation warm up, just as DEvol does, however mDEvol allows mutation warm up which maxes out relative to population size. This allows larger populations to maintain more consistency in the face of massive parallelism. DEvol and mDEvol are not significantly differentiated in terms of maximum fitnesses found over the trials shown. However, there are many characteristic differences between DEvol and mDEvol.

mDEvol is one tuned form of DEvol designed for increased exploration. The effect and amount of elitism is unmodified between DEvol and mDEvol trials shown. The pacing of the mutation warm up is fixed between DEvol and mDEvol, so differences in the first 12 generations (up to evaluation 240) are due to the effect of uniform crossover. We see divergence at times between mDEvol and DEvol, the net result of which appears to be a more rapid initial reproduction of fit genes, followed by greater exploration pressure resisting convergence pressure for a longer amount of evaluations.

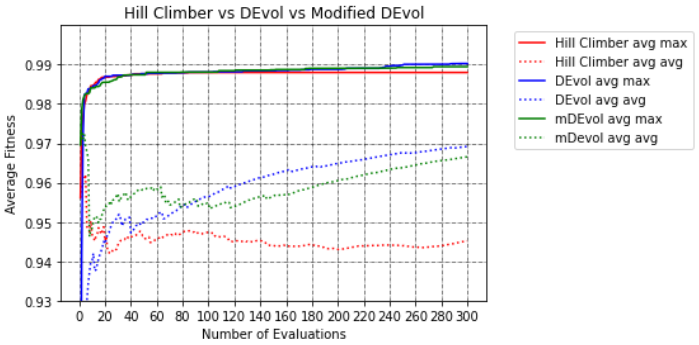

Fig. 5. Average outcomes of 10 trials each DEvol and Hill Climber, and 5 trials of mDEvol on MNIST. max_conv_layers=3, max_dense_layers=1, layer max_filters=64, max_dense_nodes=128 num_generations=15, pop_size=20, epochs=3

The time to exhaustively search the MNIST hyperparameter space using the Dsv3 shown in Table 1, using 1 epoch of training 6 convolutional layers and 2 dense layers is approximated from timing results at 8,881.44 years. This highlights the necessity of acceleration methods for the pandemic response scenario in which training time equates to inevitable loss of human life in the deployment of a viable diagnostic system. The appropriate CNN hyperparameter optimization should search through the space more efficiently than alternatives, as well as providing acceleration methods allowing for practical usage criteria to be met.

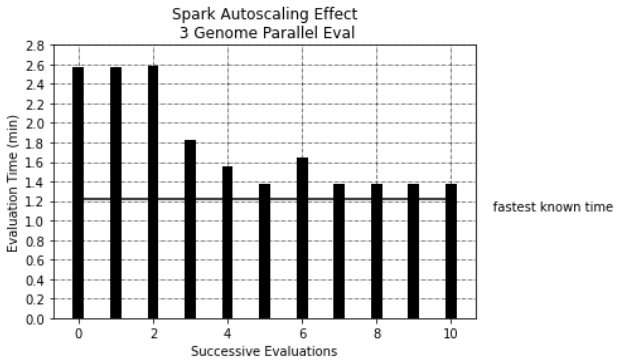

Fig. 6. Effect of Spark Autoscaling adding resources to approach the fastest known time. This can inject the data center state into timing so is disabled in all other metrics.

The Keras library employs Compute Unified Device Architecture (CUDA) drivers for GPU acceleration in data parallel fashion, for single nodes with at least one GPU [7]. This is significant because it places the bottleneck of execution, which is bandwidth between cores, in the GPU domain, which is optimized for this constraint. An alternative approach is to place this constraint upon network interface bandwidth, which can be done for parallel execution as well.

We propose a method using Apache Spark (Spark) RDDs, applying serialization to modeling components, and distributing them both through the cluster in a Directed Acyclic Graph (DAG) execution plan. Spark optimizes this DAG based on previous executions and

learns a near optimal parallel mapping between DAG components and thread pools available, by using a resource manager [6]. This allows us to evaluate training of many individuals in parallel, with a complex execution pipeline which we do not have to explicitly configure. Spark uses operational research in this area to provide both concurrency and parallel execution for data center applications [6].

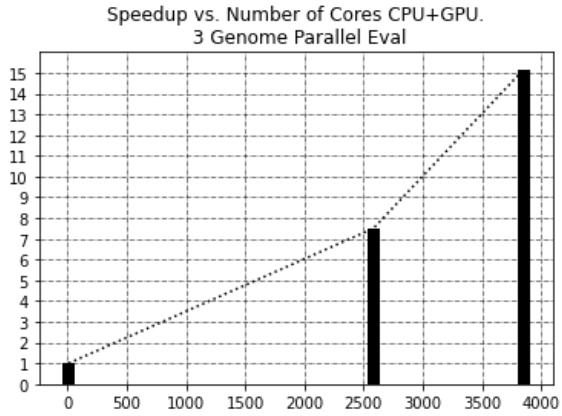

Fig. 7. Speedup vs. Number of Cores. Note the inflection, which is due to improved memory bandwidth from RDD parallelism.

We claim superlinearity in Fig !!!X due to the different speedup curves of these two methods, the combination of which is more effective than GPU data parallelism alone ($p < 0.0001$). This is demonstrated in a challenging example of only three individuals, with a total of eight convolutional and dense layers. The increasing population size of evaluation batch and model complexity both favor parallelism, as they allow more embarrassingly parallel execution. The real examples of CXR diagnostic detectors, being more complex, have much more opportunity for parallelism intrinsically at the training phase.

V Conclusion

CXR diagnostics show potential in a pandemic response context. Evolutionary computational methods may be applied to the poorly understood domain of CNN hyperparameter tuning. These evolutionary methods have a larger degree of complexity compared to gradient ascent optimizers, but represent a significant reduction of the complexity of the initial problem of hyperparameter tuning in CNNs. The improved performance of evolutionary methods is shown on a complicated example of MNIST handwriting, and tuned in this scenario. This performance tuned evolutionary hyperparameter optimizer is then adapted to a new CNN topology in SqueezeNet, resulting in Deep Squeeze Evolution. This adds fire layers to the evolutionary capability and increases the complexity of hyperparameter search.

mDEvol is shown to produce a characteristic shift in population fitness statistics compared to DEvol, which is attributable to addition of the uniform crossover operator. DEvol is further optimized for massively parallel search with addition of population scaling fitness warm up. mDEvol is shown to increase child distance from parents without loss of performance on MNIST trials. This increased disruption allows further tuning of elitist selection, mutation warm up and max, and crossover rate for high performance genetic search.

Superlinear speedup is demonstrated on state of the art cloud computing hardware. Speedup ratio of CUDA acceleration with state of the art multi GPU architecture is exceeded for cases evaluated. Spark RDD parallelization is shown to more rapidly execute three individuals' training and fitness evaluation faster than the maximum specification single node multi GPU configuration. This is traced to GPU memory bandwidth easing through DAG execution plan tuning. This has implications for any lab or group of labs doing a large amount of machine learning, wherein net efficiency of computation can be increased through use of Spark clusters.

In the context of pandemic response, precision, accuracy, speed of deployment and logistically unencumbered scalability are desirable for diagnostic tools. In a pre-clinical setting, CXR diagnostics have been demonstrated. The clinical efficacy of this method in many cases is undetermined, and is difficult to predict for future pandemics. If and when another global pandemic occurs, response and coordination may be benefited from use of evolutionary computational methods in generating diagnostic toolsets. Furthermore, these methods have the potential to improve deployment speed of testing solutions, in addition to classification performance as well. Not all encodings and layer sets have the same performance, and there is a tradeoff between complexity of the model and accuracy. An opportunity exists for case injection in these solution methods. These factors all indicate greater need for computational resources, datasets, standards and most of all clinical testing with regards to CXR diagnostic systems. Indeed an autodiagnostic case base, with associated well labeled clinical data and a GPU accelerated Spark computing environment capable of tuned genetic search all show excellent promise in the field of auto diagnostic methods for pandemic response.

N.B. Current version is draft. I will add additional work currently in progress and submit for publication early 2021. This is something I care about finishing and has been uniquely empowering during one of the most frustrating years of my life. Research has intrinsic power for reduction of suffering in the world, and in this aim I am grateful for the opportunity to pursue this topic. I have learned a lot in this experience and more than anything else I hope evidence of that is present in this draft.

References

1 WHO Coronavirus Disease (COVID-19) Dashboard https://covid19.who.int/ Accessed on 5:46pm CET, 10 December 2020

2 Beetz, C., Skrahina, V., Förster, T. M., Gaber, H., Paul, J. J., Curado, F., Rolfs, A., Bauer, P., Schäfer, S., Weckesser, V., Lieu,

V., Radefeldt, M., Pöppel, C., Krake, S., Kandaswamy, K. K., Bruesehafer, K., & Vogel, F. (2020). Rapid Large-Scale COVID-19 Testing During Shortages. Diagnostics (Basel, Switzerland), 10(7), 464. https://doi.org/10.3390/diagnostics10070464

3 Ucar, F., & Korkmaz, D. (2020). COVIDiagnosis-Net: Deep Bayes-SqueezeNet based diagnosis of the coronavirus disease 2019 (COVID-19) from X-ray images. *Medical hypotheses*, *140*, 109761. https://doi.org/10.1016/j.mehy.2020.109761

4 DEvol - Deep Neural Network Evolution https://github.com/joeddav/devol Accessed, 10 December 2020

5 LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database.

6 Salloum, S., Dautov, R., Chen, X. *et al.* Big data analytics on Apache Spark. *Int J Data Sci Anal* **1,** 145–164 (2016). https://doi.org/10.1007/s41060-016-0027-9

7 Azure Databricks pricing https://azure.microsoft.com/en-us/pricing/details/databricks/ Accessed 11 Dec 2020

8 Keras: Deep learning library for theano and tensorflow*

F Chollet. Keras: Deep learning library for theano and tensorflow (2015) https://keras. io/k

9 Matt Massie, Bernard Li, Brad Nicholes, Vladimir Vuksan, Robert Alexander, Jeff Buchbinder, Frederiko Costa, Alex Dean, Dave Josephsen, Peter Phaal, and Daniel Pocock. 2012. *Monitoring with Ganglia* (1st. ed.). O'Reilly Media, Inc.

10 Iandola, F., Han, S., Moskewicz, M., Ashraf, K., Dally, W., Keutzer, K. (2017). SQUEEZENET:AlexNET-Level Accuracy with 50X Fewer Parameters and <0.5MB Model Size. https://arxiv.org/abs/1602.07360

11 COVID-19 Image Data Collection: Prospective Predictions Are the Future. Joseph Paul Cohen and Paul Morrison and Lan Dao and Karsten Roth and Tim Q Duong and Marzyeh Ghassemi. arXiv:2006.11988, https://github.com/ieee8023/covid-chestxray-dataset, 2020

12 Baldominos, A., Saez, Y., & Isasi, P. (2018). Evolutionary Design of Convolutional Neural Networks for Human Activity Recognition in Sensor-Rich Environments. *Sensors (Basel, Switzerland)*, *18*(4), 1288. https://doi.org/10.3390/s18041288

13 Asif Iqbal Khan, Junaid Latief Shah, Mohammad Mudasir Bhat (2020). CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images,Computer Methods and Programs in Biomedicine,Volume 196,https://doi.org/10.1016/j.cmpb.2020.105581

14 L.Wang,A.Wong,COVID-Net:ATailoredDeepConvolutionalNeuralNetworkDesignforDetectionofCOVID-19CasesfromChestRadiographyImages.arXivpreprintarXiv:2003.09871.2020Mar22.

15 K. Hazelwood *et al*., "Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective," *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Vienna, 2018, pp. 620-629, doi: 10.1109/HPCA.2018.00059.

16 Eshelman (1991). The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination

17 Sushil J Louis, Gong Li, Case injected genetic algorithms for traveling salesman problems, Information Sciences, Volume 122, Issues 2–4, 2000, Pages 201-225, ISSN 0020-0255