Enhancing Neural Architecture Search with Speciation and Inter-Epoch Crossover

Matt Baughman* and Justin Wozniak (Advisor)**

*University of Chicago, Chicago, IL
**Argonne National Laboratory, Argonne, IL

***Abstract*—As deep learning continues to expand into new areas of application, the demand for efficient use of our HPC resources increase. For new problem domains, new model architectures are developed through a neural architecture search (NAS), which consist of iteratively training many neural networks. To combat the computational waste and maximize compute efficiency for NAS, we demonstrate that the use of genetic algorithms with speciation can be used to both shorten training time and increase accuracy at each iteration.**

***Index Terms*—Neural Networks, Optimization, Algorithms, Parallel Computing**

I. Background

While evolutionary algorithms have previously been used to train and optimize neural networks [1], [2], these algorithms have not been widely used in the efficient initialization and parameterization of neural networks. Efforts in neural architecture searches are largely focused on model architecture and training time for a single model [3]. However, little work has been done to use trained parameters from each run to inform the initialization of subsequent iterations. Such initialization is usually performed randomly (i.e., the weights and biases of the model are initiated at a pseudo-random value).

Following, much of the information created from the already expensive training of many models (as seen in Fig. 1) is wasted. This process underutilizes the available information and, in doing so, overlooks potential compute savings or model improvements. Therefore, dependent selection of parameter initializations could allow for benefits in training and merging models in neural architecture search through the use of readily available information.

II. Methods

To evaluate our enhanced NAS training regime, we construct a relatively small, two layer dense neural network. For data, we use the "Fashion-MNIST" dataset from Zolando [4].

We developed two workflows, one as a control and one as a test case. The control workflow (Fig. 2a) trains our neural network from N different initialization and then choses the trained model with the lowest loss after K epochs. This is equivalent to retraining the network from scratch K times to find a trained model with the best parametrization.

Our experimental workflow (Fig. 2b) begins in the same manner as the control workflow. However, during the training (for example, after $K/2$ epochs), optimization is halted and the neural networks are selectively filtered by performance. We

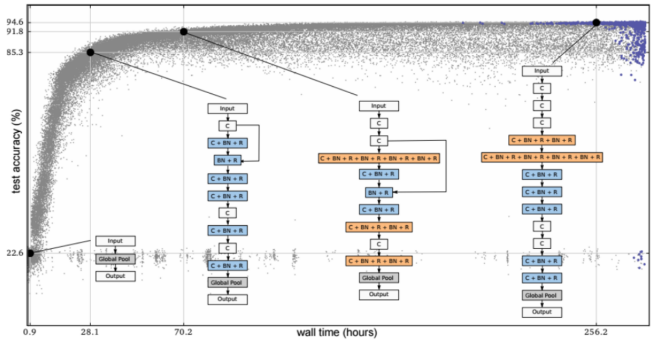
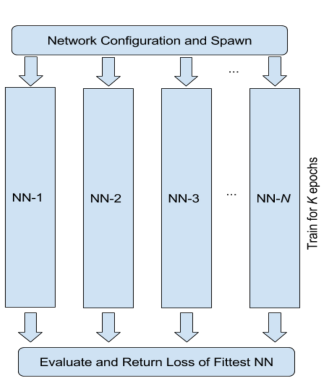


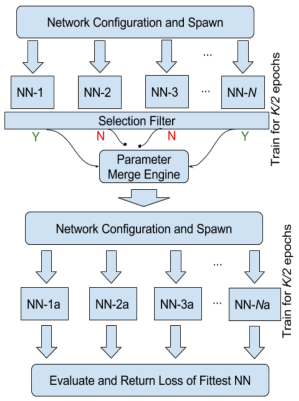
Fig. 1: Neural architecture search become exponentially expensive with respect to marginal accuracy (Image credit: Kordik, 2019)

then merge the weights of pairs of two networks using some predefined cross-over or merge function (Fig. 3). We then re-initialize N networks with slight random mutations between them and continue training until we have reached K epochs.

The neural networks were constructed using tensorflow.keras and each workflow was run 10 times in order to ensure robustness of results.



(a) Control, no cross-over.



(b) Test, with cross-over.

Fig. 2: Generalized control and test workflows used to evaluate parameter merging.

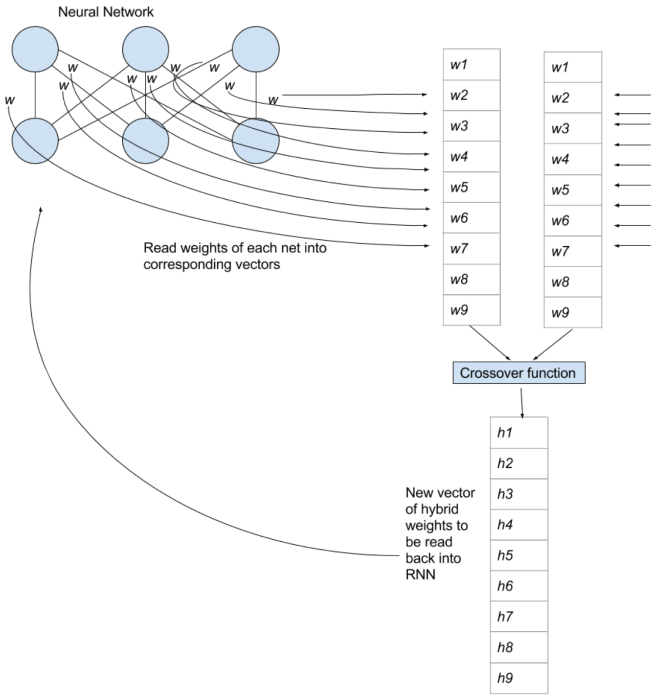


Fig. 3: Illustration of merging network weights

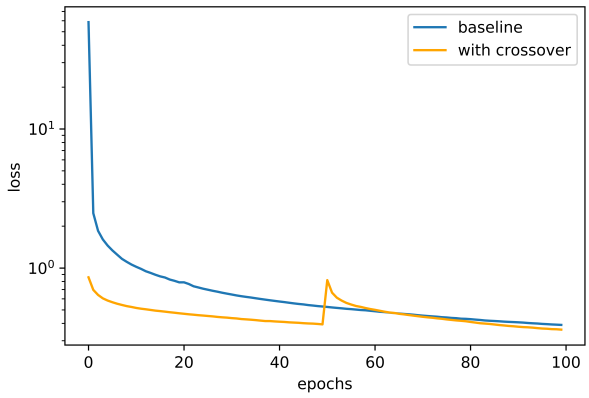


Fig. 4: Mean loss across 10 trial runs comparing the use of cross-over.

III. Results

Statistical significance was achieved between our test case and our baseline regardless of whether we used random selection or parameter means as our crossover function. Between these two methods, taking the mean of parameters showed better accuracy, though not significantly so compared to random selection. The mean improvement against the baseline after 100 epochs using the inner mean of parameters was 9%. However, results varied between workflow runs and we observed differences in effect from an increase in the ending loss by as much as 5% to a decrease the ending loss as much as 20% with respect to the control.

There is significant improvement (including a quicker convergence) following a merge of two well-trained networks (Fig 4.). This demonstrates the applicability of this approach to neural architecture search in that it allows information transfer from the networks trained throughout the search process.

Though overall results were significant, we have observed bias in our testing towards partially trained models with similar parameter values. This aligns with our initial thoughts regarding the structure and shape of the parameter search space and demonstrates the need for speciation. More specifically, for two local optima, it is not unlikely that another local optimum exists in a location between the two prior values with respect to the parameter space through which we are merging.

IV. Conclusion

The introduction of variation using fitness selection and crossover allows for quicker convergence and greater accuracy of certain neural networks. In our tests, we observed cases where little to no improvement was seen and we observed cases where error could be reduced as much as 20%. This seems to indicate dependence on initial condition and on specific network initializations and randomizations performed at the beginning of each workflow. In all, we have demonstrated the significant benefit using this method could have during the neural architecture search process by reducing the training time necessary at each generation by more than half and by reducing loss more quickly than through random network initializations.

Acknowledgments

We must thank the Illinois Institute of Technology and their BigDataX REU (NSF award 1461260), as this was the source of funding for this research. We must also thank Argonne National Laboratory for hosting this research and for providing countless hours of compute time.

References

[1] Stanley, K. and R. Miikkulainen, "Efficient evolution of neural network topologies." *Congress on Evolutionary Computing*, 2002.

[2] Wozniak, J., R. Jain, P. Balaprakash, J. Ozik, N. Collier, J. Bauer, F. Xia, T. Bretting, R. Stevens, J. Mohd-Yusof, C. Cardona, B. Essen, and M. Baughman, "Candle/supervisor: A workflow framework for machine learning applied to cancer research," *BMC Bioinformatics*, vol. 19, 2018.

[3] Zoph, B. and Q. Le, "Neural architecture search with reinforcement learning," *Proceedings of ICLR*, 2017.

[4] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[5] Amazon, "Amazon ec2 spot instances," 2019. [Online]. Available: https://aws.amazon.com/ec2/spot/

[6] P. Kordik, "Automl for predictive modeling," 2019. [Online]. Available: https://towardsdatascience.com/automl-for-predictive-modeling-32b84c5a18f6

[7] J. Aungiers, "Lstm neural network for time series prediction." 2018. [Online]. Available: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

[8] Baughman, M., C. Haas, R. Wolski, K. Chard, and I. Foster, "Predicting amazon spot prices with lstm networks," *Proceedings of the 9th Workshop on Scientific Cloud Computing*, 2018.