Search Tree Pruning for Progressive Neural Architecture Search

Deanna Flynn | deannaflynn@gci.net

UNIVERSITY of ALASKA ANCHORAGE

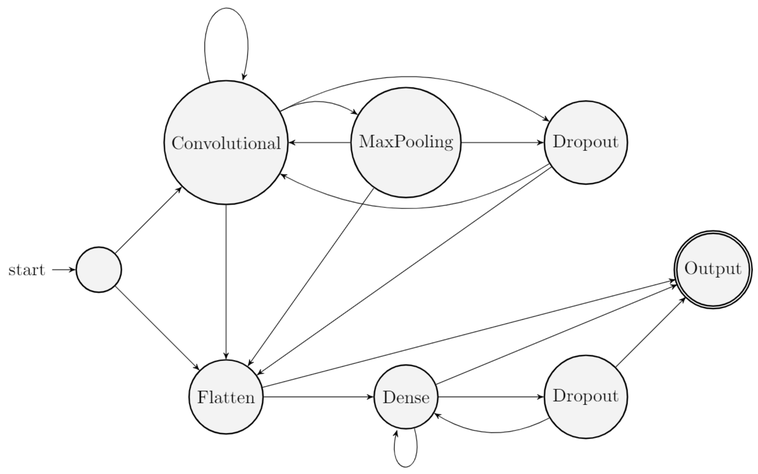
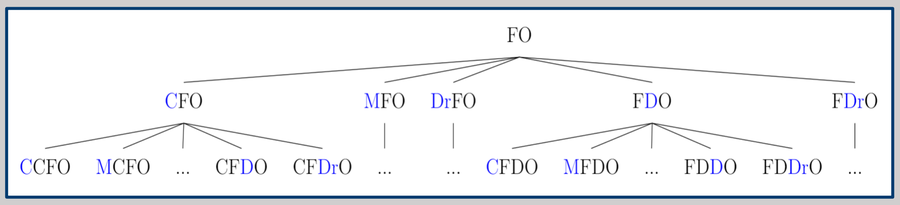


Figure 1: This transition graph defines valid neural network layer sequences in the search tree. *Output* is a 10-neuron, *Dense*, softmax layer.



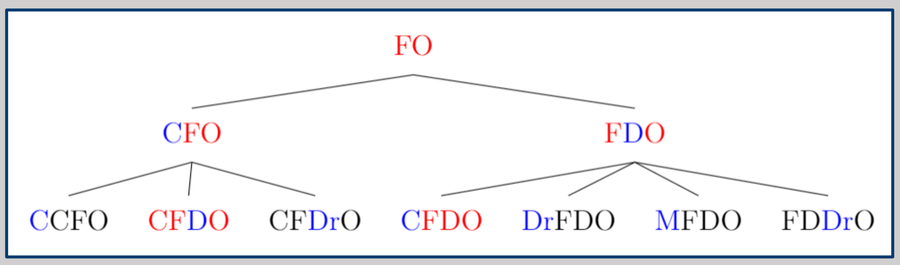


Figure 2 (top): An example search tree with new layers-***C***onvolution, ***D***ense, ***F***latten, ***Dr***opout, ***M***axPooling, and ***O***utput-in blue .
Figure 3 (bottom): An example search tree showing multiple paths leading to the same architecture (red) given the addition of new layers.

Background

Neural networks are increasing in use over the last decade, but such networks require human experts to define a specific architecture to generate optimal results. However, since the results of the network can be quantified, the search for better networks can be automated. There are several methods of automatically finding architectures of neural architectures, like Neural Architecture Search (NAS) by Zoph and Le [2], the but the algorithm explored in this poster uses a tree structure which will stop when a satisfactory architecture is created.

Progressive Neural Architecture Search Algorithm

Our algorithm is a progressive neural architecture search derived from Levin search [Schmidhuber, 1997]. Depth First Search (DFS) is used in the generation of the search tree. We expand the search tree by adding new layers into leaf-node networks up to a maximum depth. Networks which do not perform better than their parent in accuracy after training for a fixed number of epochs are removed (pruned) from the search tree. This makes our algorithm greedy in nature.

Child nodes inherit the layers, weights, and hyperparameters of their parents. New layers are drawn from the transition graph (Figure 1) given an insertion point within the parent network and the new layers' hyperparameters are optimized from a predefined list of values. Multiple networks of the same architecture can be generated but contain varying hyperparameter values. Figures 2 and 3 shows an example search tree with additional layers added in blue and the same architecture paths in red.

Transition Graph

One of the most important steps of the algorithm is the selection of a layer. Every layer within a model must be appropriate to follow the previous layer. This entails the algorithm must eliminate having, for example, a *Dropout* layer next to another *Dropout* layer, or a *MaxPooling* layer next to a *Dense* layer. Figure 1 shows a transition graph used in the addition of network layers. The edges of the graph were chosen by surveying neural network literature. An extra state called *Start* is added to signal which layers are most appropriate to begin the neural network architecture, and *Output* represents a 10-neuron, *Dense*, softmax layer.

Experiments

After four days of running on an Intel i7 8th generation CPU, 61 different network architectures were created with the best having an accuracy of 91.9% on the Fashion-MNIST dataset. Figure 4 compares our algorithm's best accuracy to other networks trained on the same dataset. Our model has <2% difference in accuracy compared to GoogleNet, the best non-preprocessed model with an accuracy of 93.7% [1]. Finally, our algorithm produced a network with less parameters in less time on a CPU compared to other benchmark models-VGG16, GoogleNet, and AlexNet. This can be seen in Table 1.

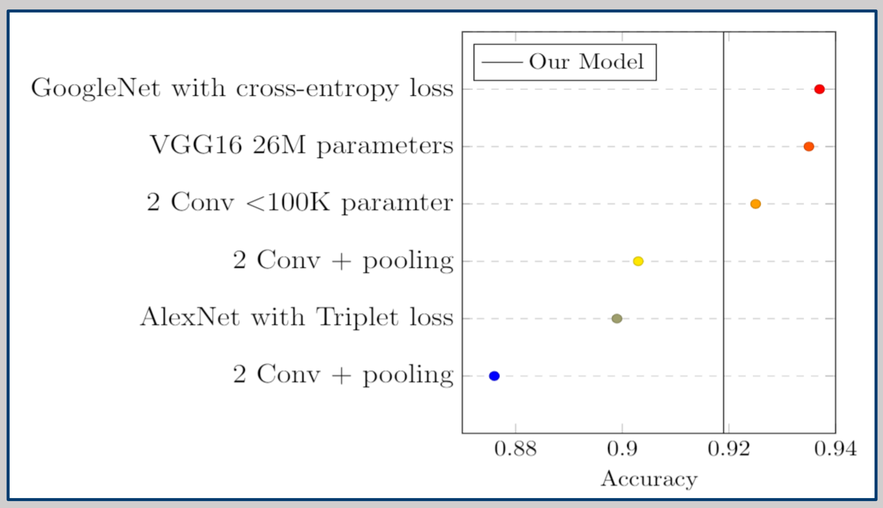


Figure 4: Accuracy of non-preprocessed model vs our model on Fashion-MNIST [Xiao, Rasul, and Vollgraf, 2017].

Future Work

In the future, varying pruning and algorithms will be implemented to compare performance of the generated networks as well as time. Adding Bayesian optimization for hyperparameter selection to speed up the algorithm.

Acknowledgements

I would like to thank my NASA mentors P. Michael Furlong and Brian Coltin as well as the Alaska Space Grant and NGA for sponsoring my internship.

Model	# of Parameters	Time to Train	Computer Specification
GoogleNet	4M		CPU
VGG16	~26M	Weeks	Nvidia Titan Black GPUs
AlexNet	~60M	6 days	2 Nvidia GeForce 580 GPUs
Our Model	**98K**	**4 days**	**Intel i7 8th Generation CPU**

Table 1: Comparison of models based on the number of parameters, the time to train, and the computer specification used to train the model. All entries for the models are based on the papers.

References

[1] Han Xiao, Kashif Rasul, Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.
[2] Barret Zoph and Quoc V le. Neural architecture search with reinforcement learning. *ICLR*, 2017.
[3] Schmidhuber, J. 1997. Discovering neural nets with low kolmogorov complexity and high generalization capability. Neural Networks 10(5):857–873
[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
[5] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
[6] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).