NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithms Supplementary Materials

Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar,
Kalyanmoy Deb, Erik Goodman and Wolfgang Banzhaf
Michigan State University
East Lansing, Michigan
{luzhicha,whalenia,vishnu,dhebarya,kdeb,goodman,banzhafw}@msu.edu

ACM Reference Format:
Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman and Wolfgang Banzhaf. 2019. NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithms Supplementary Materials. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3321707.3321729

1 DUPLICATE CHECKING AND REMOVAL

Due to the directed acyclic nature of our encoding, redundancy exists in the search space defined by our coding, meaning that there exist multiple encoding strings that decode to the same network architecture. Empirically, we have witnessed the redundancy becomes more and more severe as the allowed number of nodes in each phase's computational block increase, as shown in Figure 1.

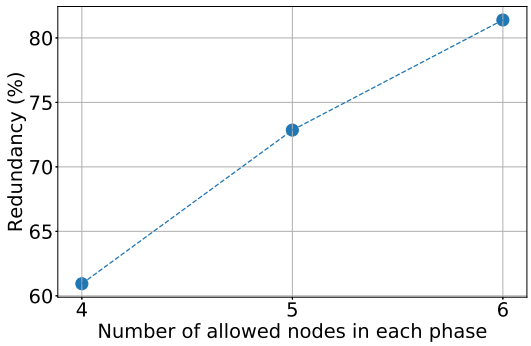


Figure 1: Increase in redundancy as node count increases.

Since the training of a deep network is a computationally taxing task, it is essential to avoid the re-computation of the same architecture. In this section, we will provide with an overview of an algorithm we developed to quickly and approximately do a *duplicate-check* on genomes. The algorithm takes two genomes to be compared as an input, and outputs a *flag* to indicate if the supplied genomes decode to same architecture.

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321729

In general, comparing two graphs is NP-hard, however, given that we are working with Directed Acyclic Graphs with every node being the same in terms of operations, we were able to design an efficient network architecture duplicate checking method to identify most of the duplicates if not all. The method is built on top of simply intuition that under such circumstances, the duplicate network architectures should be identified by swapping the node numbers. Examples are provided in Figure 2. Our duplicates checking method first derive the connectivity matrix from the bit-string, which will have positive 1 indicating there is an input to that particular node and negative 1 indicating an output from that particular node. Then a series row-and-column swapping operation takes place, which essentially try to shuffle the node number to check if two connectivity matrix can be exactly matched. Empirically, we have found this method performs very efficiently in identifying duplicates. An example of different operation encoding bit-strings decode to the same network phase is provided in Figure 2.

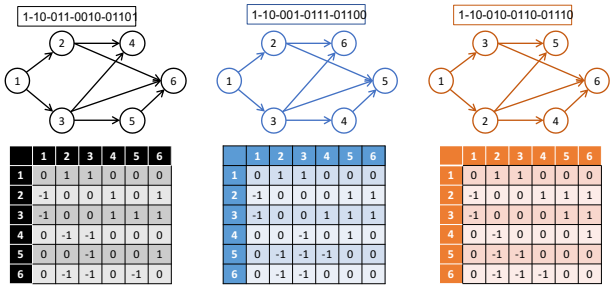
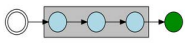
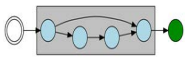
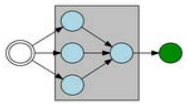
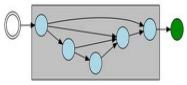
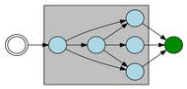
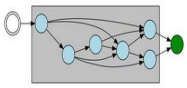


Figure 2: Examples of different encoding bit strings that decode to the same network computation block.

2 ARCHITECTURE COMPLEXITY ESTIMATION

We argue that the choice of inference time or number of parameters as proxies for computational complexity are sub-optimal and ineffective in practice. In fact, we initially considered both of these objectives. We concluded from extensive experimentation that inference time cannot be estimated reliably due differences and inconsistencies in computing environment, GPU manufacturer, and GPU temperature etc. Similarly, the number of parameters only relates one aspect of computational complexity. Instead, we chose to use the number of floating-point operations (FLOPs) for our second objective. The following table compares the number of active nodes, the number of connections, the total number of parameters and the FLOPs over a

Table 1: Network examples comparing the number of active nodes, number of connections, number of parameters and number of multiply-adds.

Phase Architectures	# of Nodes	# of Conns	Params. (K)	FLOPs (M)
	3	4	113	101
	4	6	159	141
	4	7	163	145
	5	9	208	186
	5	10	216	193
	6	13	265	237

few sampled architecture building blocks. See Table 1 for examples of these calculations.

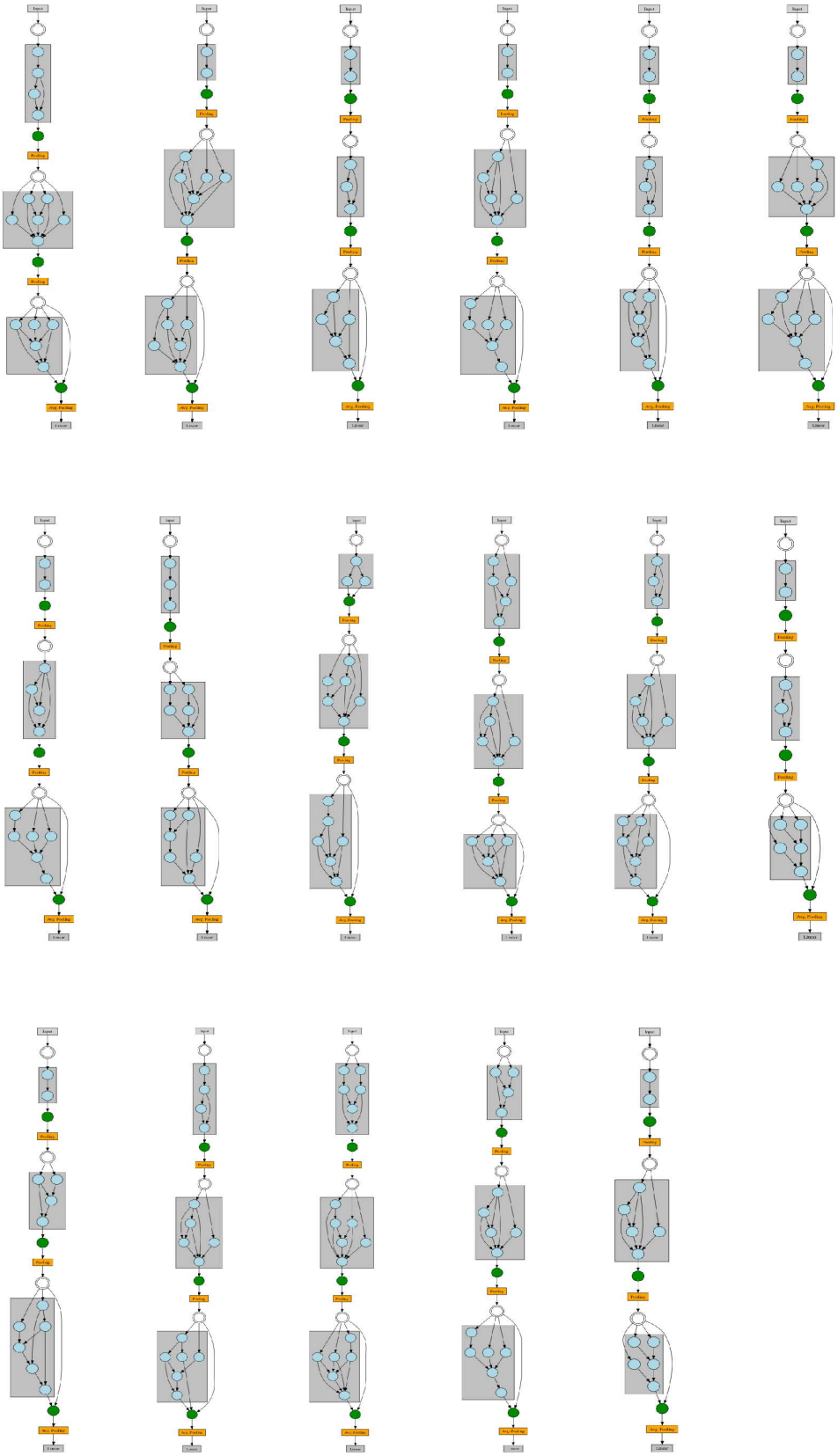


Figure 3: Set of networks architectures on the trade-off frontier discovered by NSGA-Net.

Table 2: Summary of relevant related work along with datasets each method has been applied to, objectives optimized, and the computational power used (if reported). Methods not explicitly named are presented as the author names. PTB refers to the Penn Treebank [14] dataset. The Dataset(s) column describes what datasets the method performed a search with, meaning other datasets may have been presented in a study, but not used to perform architecture search. A dash represents some information not being provided. We attempt to limit the focus here to published methods, though some unpublished methods may be listed for historical contingency.

	Method Name	Dataset(s)	Objective(s)	Compute Used
RL	Zoph and Lee [21]	CIFAR-10, PTB	Accuracy	800 Nvidia K80 GPUs 22,400 GPU Hours
	NASNet [22]	CIFAR-10	Accuracy	500 Nvidia P100 GPUs 2,000 GPU Hours
	BlockQNN [20]	CIFAR-10	Accuracy	32 Nvidia 1080Ti GPUS 3 Days
	MetaQNN [1]	SVHN, MNIST CIFAR-10	Accuracy	10 Nvidia GPUs 8-10 Days
	MONAS [7]	CIFAR-10	Accuracy & Power	Nvidia 1080Ti GPUs
	EAS [2]	SVHN, CIFAR-10	Accuracy	5 Nvidia 1080Ti GPUs 2 Days
	ENAS [16]	CIFAR-10, PTB	Accuracy	1 Nvidia 1080Ti GPUs < 16 Hours
EA	CoDeepNEAT [15]	CIFAR-10, PTB	Accuracy	1 Nvidia 980 GPU
	Real *et al.* [18]	CIFAR-10, CIFAR-100	Accuracy	-
	AmoebaNet [17]	CIFAR-10	Accuracy	450 Nvidia K40 GPUs ~7 Days
	GeNet [19]	CIFAR-10	Accuracy	10 GPUs 17 GPU Days
	NEMO [10]	MNIST, CIFAR-10 Drowsiness Dataset	Accuracy & Latency	60 Nvidia Tesla M40 GPUs
	Liu *et al.* [12]	CIFAR-10	Accuracy	200 Nvidia P100 GPUs
	LEMONADE [6]	CIFAR-10	Accuracy	Titan X GPUs 56 GPU Days
	PNAS [11]	CIFAR-10	Accuracy	-
	PPP-Net [5]	CIFAR-10	Accuracy & Params/FLOPS/Time	Nvidia Titan X Pascal
Other	NASBOT [9]	CIFAR-10 Various	Accuracy	2-4 Nvidia 980 GPUs
	DPC [3]	Cityscapes [4]	Accuracy	370 GPUs 1 Week
	NAO [8]	CIFAR-10	Accuracy	200 Nvidia V100 GPUs 1 Day
	DARTS [13]	CIFAR-10	Accuracy	1 Nvidia 1080Ti GPUs 1.5 - 4 Day

REFERENCES

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2017. Designing Neural Network Architectures using Reinforcement Learning. In *ICLR*.

[2] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Efficient Architecture Search by Network Transformation. In *AAAI*.

[3] L.-C. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. 2018. Searching for Efficient Multi-Scale Architectures for Dense Image Prediction. *arXiv preprint arXiv:1809.04184* (Sep 2018). arXiv:cs.CV/1809.04184

[4] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. 2014. Detect What You Can: Detecting and Representing Objects Using Holistic Models and Body Parts. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1979–1986. https://doi.org/10.1109/CVPR.2014.254

[5] Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. 2018. PPP-Net: Platform-aware Progressive Search for Pareto-optimal Neural Architectures. In *ICLR*.

[6] T. Elsken, J. Hendrik Metzen, and F. Hutter. 2018. Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution. *arXiv preprint arXiv:1804.09081* (April 2018). arXiv:stat.ML/1804.09081

[7] C.-H. Hsu, S.-H. Chang, D.-C. Juan, J.-Y. Pan, Y.-T. Chen, W. Wei, and S.-C. Chang. 2018. MONAS: Multi-Objective Neural Architecture Search using Reinforcement Learning. *arXiv preprint arXiv:1806.10332* (June 2018). arXiv:1806.10332

[8] C.-H. Hsu, S.-H. Chang, D.-C. Juan, J.-Y. Pan, Y.-T. Chen, W. Wei, and S.-C. Chang. 2018. Neural Architecture Optimization. *arXiv preprint arXiv:1808.07233* (Aug 2018). arXiv:1808.07233

[9] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. Xing. 2018. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. *arXiv preprints arXiv:1802.07191* (Feb 2018). arXiv:1802.07191

[10] Y.H. Kim, B. Reddy, S. Yun, and C. Seo. 2017. NEMO: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy. In *JMLR: Workshop and Conference Proceedings*, Vol. 1. 1–8.

[11] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. 2017. Progressive Neural Architecture Search. *CoRR* abs/1712.00559 (2017). arXiv:1712.00559 http://arxiv.org/abs/1712.00559

[12] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. 2018. Hierarchical Representations for Efficient Architecture Search. In *ICLR*. https://openreview.net/forum?id=BJQRKzbA-

[13] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable Architecture Search. *arXiv preprint arXiv:1806.09055* (2018).

[14] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the Workshop on Human Language Technology (HLT '94)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 114–119. https://doi.org/10.3115/1075812.1075835

[15] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. 2017. Evolving Deep Neural Networks. *arXiv preprint arXiv:1703.00548* (March 2017). arXiv:1703.00548

[16] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameter Sharing. *CoRR* abs/1802.03268 (2018). arXiv:1802.03268 http://arxiv.org/abs/1802.03268

[17] E. Real, A. Aggarwal, Y. Huang, and Q. V Le. 2018. Regularized Evolution for Image Classifier Architecture Search. *arXiv preprint arXiv:1802.01548* (Feb 2018). arXiv:1802.01548

[18] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin. 2017. Large-Scale Evolution of Image Classifiers. *arXiv preprint arXiv:1703.01041* (March 2017). arXiv:1703.01041

[19] L. Xie and A. Yuille. 2017. Genetic CNN. In *ICCV*.

[20] Zhao Zhong, Junjie Yan, and Cheng-Lin Liu. 2017. Practical Network Blocks Design with Q-Learning. *CoRR* abs/1708.05552 (2017). arXiv:1708.05552 http://arxiv.org/abs/1708.05552

[21] B. Zoph and Q. V. Le. 2016. Neural Architecture Search with Reinforcement Learning. *arXiv preprint arXiv:1611.01578* (Nov 2016). arXiv:cs.LG/1611.01578

[22] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8697–8710.