Evolutionary Selection Criteria and Performance in NAS-Bench-101

Jordan Donovan
University of Vermont
Burlington, VT
jdonova6@uvm.edu

1 PROBLEM STATEMENT

Artificial neural networks have recently become one of the best methods of discovering and extrapolating patterns in data, especially for predictive purposes ([2], [3], [1], [18], [11], [15], [10]). The structure, or architecture, of these networks contributes greatly to their success and is critical in the discovery of these patterns ([14]). Historically, these neural network architectures were hand-made by human designers [[11], [2], [3]], which is a time consuming and costly task.

More recently, an interest has started to grow in the idea of automatically generating these architectures (neural architecture search, or NAS) using various techniques ([9], [12], [20], [13], [5], [17]). Some of these techniques have made use of building blocks to construct these neural architectures which involve searching for an inner structure for each block to maximize performance of the architecture on some task. ([12], [19], [13], [17]). These methods are still constrained by the objective-driven human designers and the selection operators invoked to search the space of these inner structures.

It is hypothesized that utilizing more exploratory selection operators will allow for evolutionary NAS to access more diverse solutions resulting in potentially higher performance of solutions. This work will explore this concept within the realm of the NAS-Bench-101 dataset and will apply to the research direction of the student's dissertation by portraying the limiting nature of current inner cell selection operators in NAS and showing the power of exploratory ones.

2 RELATED WORK

Related Work: Neural networks have come to excel at extracting complex patterns from data to solve complex, nonlinear problems ([2], [3], [1], [18], [11], [15], [10]]. The architectural designs of these networks in many cases are derived manually by human creators ([11], [2], [3]).

More recently, a technique called neural architecture search, which has the goal of automating the construction of these architectures, has shown much promise ([9], [12], [20], [13], [5], [17]). In this area of research, two main techniques for discovering these architectures extrapolate knowledge. The first method is differential neural architecture search ([20], [13], [5]), and the second is evolutionary neural architecture search ([9], [7], [8], [12]). All of these methods are very highly constrained in their search for competitive architectures and much of the design is largely biased because it is created by a human designer ([9], [20], [13], [5]].

One of the growingly popular techniques in NAS, building blocks with inner structures, has shown its capability at producing complex architectures and performing well on various tasks ([12], [13], [20], [7], [17]).

This work utilizes techniques from recent successful approaches to NAS, specifically the NAS-Bench-101 search space and dataset which involves the building blocks mentioned above ([17]).

This work also takes inspiration from prior research in exploration and diversity maintenance in regards to evolutionary algorithms. The concept of novelty search has proven to be a successful approach to discovery within evolutionary processes by incentivizing novel behavior / genetics instead of fitness on the target task ([4]).

Other techniques of diversity maintenance include:

- increasing population size,
- adding random restarts,
- adding noise to mutations,
- invoking less selection pressure,
- adding explicit separation (such as island models),
- explicitly trying to add individuals that are unique compared to the rest of the population,
- fitness sharing,
- and crowding.

The author takes this list from its enumeration in [6].

Increasing the population size, increases the diversity by allowing for potentially more comprehensive coverage of the space of possible solutions since there are more individuals, thus, potentially, more diverse individuals.

Adding random restarts, allows for more diversity in a similar manner by allowing for the restarts to potentially result in more exploration of the space of solutions since the restarts might start the population in diverse locations in the search landscape.

Adding noise to mutations makes it possible for more novel alleles to emerge in the population which increases diversity.

Less selection pressure allows for individuals that are diverse from top-performing solutions, and that might otherwise by thrown out due to low fitness scores, to mutate which maintains the diversity of the population.

Adding explicit separation, or niching, allows for diverse subpopulations to form and compete within themselves as well as between groups.

CS 352: Final Projects, Dec. 7–14, 2021, UVM

It is simple to see diversity being maintained when explicitly adding diversity via adding unique individuals to a population that has started to lose diversity.

Fitness sharing lowers the fitness of individuals whose fitness is too close to others in the population allowing for individuals with different fitness values to survive.

Crowding kills off individuals that are too similar to incoming children eliminating early convergence and allowing for more diversity.

3 METHODOLOGY

This work implements an evolutionary algorithm to evolve inner-cell solutions within the NAS-Bench-101 space ([17]). These inner-cells involve up to 7 nodes with up to 9 edges. They must include at least the input node, output node, and two hidden nodes. Once the inner cell is formed, it is then stacked thrice to form a "stack" which is then, again, stacked thrice with downsampling between them to form the final architecture of the network. More details about the dataset (such as training hyperparamters, architecture assembly, and included metrics) can be found in the original paper ([17]).

Various evolutionary selection operators are implemented for the purpose of in-depth investigation. These include:

- age-based ([7]),
- fitness,
- novelty ([4]),
- mixed novelty-fitness (50% / 50%),
- and random

The evolutionary algorithm in which each of these techniques is implemented is constant and does not change outside of the selection process.

Two code repositories are utilized in this work. The first repository is that of the NAS-Bench-101 dataset[1] ([16]). It also provides tools for detecting isomorphisms in graphs and fetching accuracies of networks at different epoch budgets and on both validation and test sets of CIFAR-10. The second repository is that of the benchmarking code for NAS-Bench-101[2] ([17]).

Utilizing the NAS-Bench-101 dataset and benchmarking repositories allows for the evolved inner cells' fitness values, which correspond to accuracy on CIFAR-10, to be obtained quickly and simply. This enables much more efficient comparisons and experiments when compared to other NAS techniques that involve not only creating, comparing, and mutating networks (as done here) but additionally training and evaluating them. It also allows for direct comparison with other work utilizing the NAS-Bench-101 dataset. With fitness values being so easily obtained, the novelty computation is the most time-consuming piece of the algorithms.

Novelty is measured by first storing the population from the previous generation in a history variable and second by iteratively comparing each of the inner cell networks of the current population to every member of both the current population and the previous population. This comparison involves several steps. First, the network being observed is converted into a directed graph, then an iterative approach is taken to remove all vertices, and their edges, in this graph that do not connect to both input and output since they are computationally irrelevant. Once this final, concise graph is constructed, the adjancency matrix of it is used to retrieve the lambda distance between the graph comparisons mentioned above. The average lambda distance across all comparisons is taken as the novelty metric of each particular inner cell network.

Each evolutionary algorithm is run 20 times with the following hyperparameters:

- population size = 50,
- number of children = 50,
- tournament size = 10,
- number of winners = 5,
- and generations = 100.

The highest fitness value for each generation, in each run, is recorded as is the average novelty of all individuals in the population. The fitness and novelty values are then averaged across the 20 runs for each selection criteria. All of the results for diversity and accuracy of the models produced from these evolutionary algorithms is then compared.

These results are unique to the field of NAS, since, to the author's knowledge, this type of comparison has not been directly addressed in the literature, especially regarding this specific dataset.

4 RESULTS

One of the main results is a comparison of the highest fitness values across generations for each of the selection types listed above. The average of the 20 runs for each of the selection types, except novelty, is shown in Figure 1. The reason we do not include novelty search in this fitness plot can be seen in Figure 2. Novelty does not perform well in comparison to the other techniques when it comes to the highest fit individual at each generation which flattens the other selection criteria making it difficult to interpret the plot. The results show that fitness-based evolution performs the best in regards to this metric, with mixed novelty-fitness as a close second. Even random search does fairly will in this metric.

The other main result is a comparison of the average novelty values of the population across generations for each of the selection types. Again, the average of 20 runs is used for each of the selection types. The results are plotted in Figure 3 with a 95% confidence interval. These results seem to follow the intuitions of the author. Selecting for novelty results in the highest novelty, and selecting for mixed novelty and fitness (50%, 50%) results in half as much novelty. Interestingly, fitness results in a significant amount more novelty than random search or regularized evolution.

The most interesting of these results is that the mixed novelty and fitness evolutionary algorithm maintains a much higher level of novelty compared with all of the other techniques, save novelty selection, but also results in fitness values that are comparable with that of the fitness-based selection, which had the best results in these experiments with regards to finding fit individuals.

5 DISCUSSION

Comparing selection criteria along the axes of novelty and fitness in the NAS-Bench-101 search space is a first step to applying it more broadly to NAS. The results above show that in this particular case, novelty alone is not an optimal selection metric when

[1] https://github.com/automl/nasbench
[2] https://github.com/automl/nas_benchmarks

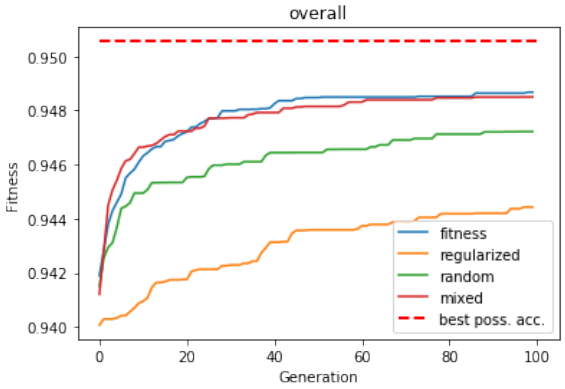

Figure 1: Average of highest fit individual for each generation across 20 runs for each selection type, except novelty.

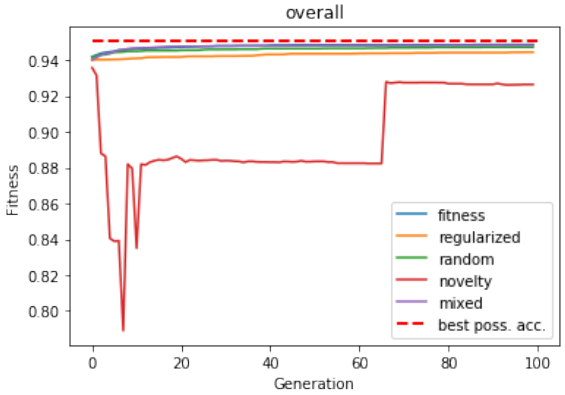

Figure 2: Average of highest fit individual for each generation across 20 runs for each selection type. One can see why novelty is excluded in 1

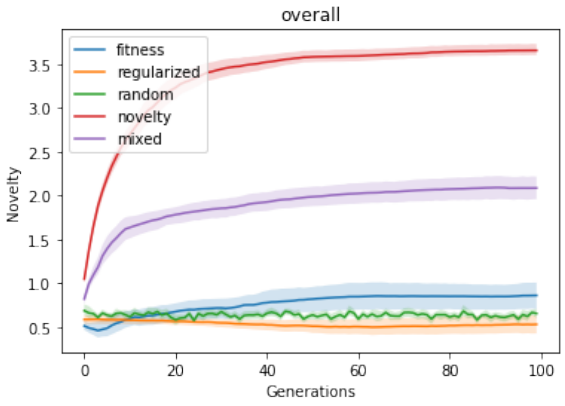

Figure 3: Average, with 95% CI, of the average novelty of the population at each generation across 20 runs for each selection type.

considering fitness of solutions. This is not surprising since the original NAS-Bench-101 work explored the diversity of solutions with high fitness, concluding that they were all within a relatively close range of the solution space ([17]). For this reason, the results obtained here are still valid and apply to other search spaces with this characteristic, but these results may not carry as much weight in domains with differing characteristics.

The author believes that the reason that mixed novelty and fitness does so well here is because of its particular implementation in the evolutionary algorithm. In the tournament selection loop of this particular selection criteria, the tournament winners are selected randomly either as the most novel or the most fit. Additionally, the selection after mutations on the children, is completed by selecting the top 25 (half of the population size) most fit, and the top 25 most novel individuals in the population. This allows for the population to maintain both a diverse set of solutions as well as the most fit solutions.

Another consideration with regard to these experiments is the novelty metric itself. Each inner cell network being evolved consists of operations and edges per the original work ([17]). The novelty metric chosen for the experiments is a lambda distance of the adjancency matrices of the networks converted to graphs. This lambda distance does not take into account the operations on the nodes or the order of the nodes. This could result in potentially higher, or lower, novelty scores than would be awarded if these were taken into account.

Finally, the author notes that the original work ([17]) utilizes regularized evolution with an order of magnitude more iterations than this work allows. Regularized evolution is much less computationally costly than some of the other methods here. If allowed to have a higher generation count, it might compete more in the metrics recorded here.

6 FUTURE WORK

Further experimentation along the lines of this work could yield interesting results. The author notes a few of these suggestions here.

As mentioned in the previous section, the results of this work may not transfer or be meaningfully relevant to NAS when the search space is significantly different. For instance, if the search space were to be more rugged and less localized, these experiments could yield very different results. It is suggested that these selection criteria and others be applied to other NAS search spaces such as the NASNet search space.

Additionally, the novelty metric utilized here could be updated to use something that takes both the operations and the structure of the inner cells into account. This could change the results significantly if these experiments exploited the method of novelty calculation used here.

Further experiments that allow regularized evolution to utilize the same computational budget as other selection criteria could be performed since it has been noted that in the past this has shown some benefit in regards to the results that this technique can produce.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Nicholas Cheney for providing feedback throughout this process.

The authors would also like to thank the entire class of CS 352 - Evolutionary Computation for their valuable comments and helpful suggestions.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova Google, and A I Language. [n. d.]. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ([n. d.]). arXiv:1810.04805v2 https://github.com/tensorflow/tensor2tensor

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. [n. d.]. Deep Residual Learning for Image Recognition. ([n. d.]). arXiv:1512.03385v1 http://image-net.org/challenges/LSVRC/2015/

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. [n. d.]. Densely Connected Convolutional Networks. ([n. d.]). arXiv:1608.06993v5 https://github.com/liuzhuang13/DenseNet.

[4] Joel Lehman and Kenneth O Stanley. 2011. Abandoning Objectives: Evolution through the Search for Novelty Alone. *Evolutionary Computation journal* 19 (2011), 189–223.

[5] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, † Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. [n. d.]. DARTS+: Improved Differentiable Architecture Search with Early Stopping. ([n. d.]). arXiv:1909.06035v2

[6] Sean Luke. 2016. Essentials of Metaheuristics A Set of Undergraduate Lecture Notes by Second Edition. (2016), 127–128. http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/

[7] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*. 4780–4789. https://doi.org/10.1609/aaai.v33i01.33014780 arXiv:1802.01548

[8] Esteban Real, Chen Liang, David R So, and Quoc V Le. 2020. AutoML-Zero: Evolving Machine Learning Algorithms From Scratch. (2020). https://github.

[9] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-Scale Evolution of Image Classifiers. (2017). arXiv:1703.01041v2

[10] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016). https://doi.org/10.1038/nature16961

[11] Karen Simonyan and Andrew Zisserman. 2015. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. (2015). arXiv:1409.1556v6 http://www.robots.ox.ac.uk/

[12] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Jiancheng Lv. [n. d.]. Automatically Designing CNN Architectures Using Genetic Algorithm for Image Classification. ([n. d.]). https://doi.org/10.1109/TCYB.2020.2983860 arXiv:1808.03818v3

[13] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. [n. d.]. MnasNet: Platform-Aware Neural Architecture Search for Mobile. ([n. d.]). arXiv:1807.11626v3 https://github.com/tensorflow/tpu/

[14] Gerhard Weiß. [n. d.]. Neural Networks and Evolutionary Computation. Part I: Hybrid Approaches in Artificial Intelligence. ([n. d.]).

[15] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. [n. d.]. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. ([n. d.]). arXiv:1609.08144v2

[16] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. 2019. NAS-Bench-101: Towards Reproducible Neural Architecture Search. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, Long Beach, California, USA, 7105–7114. http://proceedings.mlr.press/v97/ying19a.html

[17] Chris Ying, Aaron Klein, Esteban Real, Eric Christiansen, Kevin Murphy, and Frank Hutter. [n. d.]. NAS-Bench-101: Towards Reproducible Neural Architecture Search. ([n. d.]). arXiv:1902.09635v2

[18] Yu Zhang, William Chan, and Navdeep Jaitly. [n. d.]. VERY DEEP CONVOLUTIONAL NETWORKS FOR END-TO-END SPEECH RECOGNITION. ([n. d.]). arXiv:1610.03022v1

[19] Barret Zoph, Google Brain, Vijay Vasudevan, Jonathon Shlens, Quoc V Le Google Brain, and Quoc V. Le. 2018. *Learning Transferable Architectures for Scalable Image Recognition.* Technical Report. 8697–8710 pages. https://doi.org/10.1109/CVPR.2018.00907 arXiv:1707.07012

[20] Barret Zoph, Quoc V. Le, and Quoc V Le Google Brain. 2017. *NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING.* Technical Report. arXiv:1611.01578v2