Neural Architecture Search using NASBench-101 Dataset

Akash Malhotra* Shubham Biswas†

31 January 2021

Abstract

This work presents some explorations into NASBench-101 dataset. First we present 3 search algorithms to search this dataset for the architecture that gives best test accuracy on CIFAR-10. Evolutionary search appears to be the best. Then, we show that the accuracies of the same architectures on 3 different datasets are strongly correlated. This is a surprising discovery as these 3 datasets are quite different. In this way, NASBench-101 dataset could be used as a reference point while designing a new network for a custom Image classification problem.

Keywords: AutoDL, NAS, CNN

1 Introduction

AutoDL is a branch of Deep Learning where the goal is to find the optimum architecture automatically without Human in the Loop. This requires

1. Defining the search space of the architecture
2. Finding the right algorithm to find the optimum architecture in this search space.
3. Evaluating the searched architecture.

Normally, such search spaces are defined using some inductive bias (e.g. CNNs for Image Classification) and some heuristics (e.g. architectures similar to Inception)

NASBench-101 [8] is a database of 423,624 CNN architectures evaluated on CIFAR-10 dataset for 12, 36 and 108 epochs respectively. They define the search space using cells (inception like cell) which is a repeated pattern of Data -> Convolutions -> Maxpool observed in big CNNs like Alexnet and Inception v3. CNN architecture is defined as stack of cells. For more details, see Appendix.

There have been many strategies to do efficient Neural Architecture Search for CNN. AlphaX [6] is a search strategy that is inspired by Alpha Go [5] which was designed by Deep

*BDMA, CentraleSupélec, University of Paris Saclay. Email: akash.malhotra@student.cs-fr.
†BDMA, CentraleSupélec, University of Paris Saclay. Email: shubham.biswas@student.cs-fr.

Mind to play the game of Go. In this approach, optimum architecture is searched using Monte Carlo Tree Search, which consists of 4 phases: Selection -> Expansion -> Simulation -> Backpropagation (For more info see Appendix). AlphaX uses a meta-DNN (as a policy net) to predict accuracies in the simulation and back-propagates its value throughout the tree. It also uses transfer learning for warm for the architectures that are similar to their parents.

AlphaX is quite an expensive strategy to do Neural Architecture Search. For our project, we lack the computational resources to train and use a meta-DNN, which is a multi-way neural network to do perform Monte Carlo Tree Search. Instead, we are interested if we can use the NASBench-101 dataset in a meaningful way since it already contains the accuracies of searched architecture on CIFAR-10. Thus, the main objective of this project is as follows:

1. Compare different search algorithms to search the dataset.
2. Compare accuracy of sampled architecture on different datasets.

2 Method

2.1 Set up

We ran our experiments in Google Colaboratory notebook with a single Tesla T4 GPU with 16 GB of memory. Also, for all our CNNs, we used 128 samples per batch.

2.2 Comparison of different search algorithms.

We compare performance of three search algorithms on NASBench-101 dataset. The metric is the highest validation accuracy found in 1)given time 2)for given number of nodes.

1. Random Search: We simply randomly sample a cell, query the NASBench-101 dataset and record validation and test accuracy. We store the best accuracy.

2. Evolutionary Search: Evolutionary search was performed by setting population size of 50 and tournament size of 10. Mutation was done by random combination. Similar to random search, we store the best accuracy in each iteration.

3. Tree Search: This is inspired by AlphaX. But here, since we do not use meta-DNN, we assume that the Search Tree is already built. Hence, we only do selection phase, using UCB1 [2] algorithm. We treat validation accuracy of CIFAR-10 [4] as the reward signal. The children at each node are generating by either adding/removing an edge, or by changing the node in the Cell DAG.

2.3 Insight into NASBench-101 dataset

For tree search to work in the way we are making it work, the dataset should satisfy the assumption that similar architectures yield similar accuracy. We design a distance metric between two randomly sampled architectures.

Distance between two cells is measured as follows:

We generate edge triplets (Node1, edge, Node2) for each cell, as cell is a DAG. Distance is how many tuples are in cell1 that are not in cell2, plus the size difference between cell1 and cell2. The distance is commutative.

Then we plot distance vs difference in accuracy and see if correlation is significant.

2.4 Performance of same architecture on different datasets

We are interested in finding out if the similar architectures yield similar accuracies for different datasets. We already know the test accuracies for CIFAR-10 dataset of all the possible CNNs in NASBench search space. So we randomly sample significant number of samples and train them on FashionMNIST [7] for 6 epochs, and later we also trained them on Imagenette [3] for 36 epochs. We choose FashionMNIST because:

1. It is very different from CIFAR-10 on which NASBench dataset is trained. (grayscale, about clothes)

2. It is not as simple as overused MNIST data. But simple enough to achieve good accuracy in low number of epochs. (6 epochs)

3. Images are not big in size so many neural networks can be trained with the limited resources we have.

We choose Imagenette because:

1. It is the best of ImageNet, but a much smaller version, with only 10 classes, and a CNN architecture can be tested quickly on it.

2. The CNN architecture that performs well on Imagenette tends to perform well on ImageNet. Images in Imagenette are bigger in size than in FashionMNIST. So we would need to change the architecture of CNN. To change the architecture, we just added couple of more cells to the architecture sampled for FashionMNIST. Our custom cell is of kind Input->(Conv3x3->Conv3x3 concat Conv3x3)->MaxPool2x2. We also used various data augmentation techniques and normalization.

3 Results and Discussion

3.1 Search performance

In the figures 1 and 2, we see the performance of different search algorithms. The best performing algorithm for the given amount of time is the evolutionary search, followed by random and then tree search. At first, we thought that tree search performs worse than other two because it explores less number of cells (due to tree class we created, which is quite heavy and therefore a bottleneck). Random Search on average, explores 4000 cells, Evolutionary Search explores around 3000, while tree search only explores 300 cells in 5 seconds. So, we also decided the compare the search algorithms by number of cells explored, to see if it is worth to make the code for tree search faster. In figure 2, we show

the results. Tree Search still under-performs. Figure 3 further confirms this with the mean accuracy of graph of 10 runs.

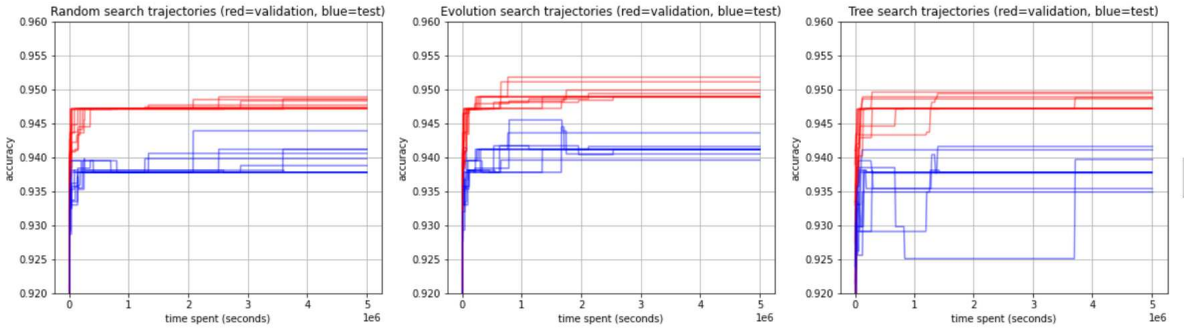

FIGURE 1: Comparison between different search algorithms by time.

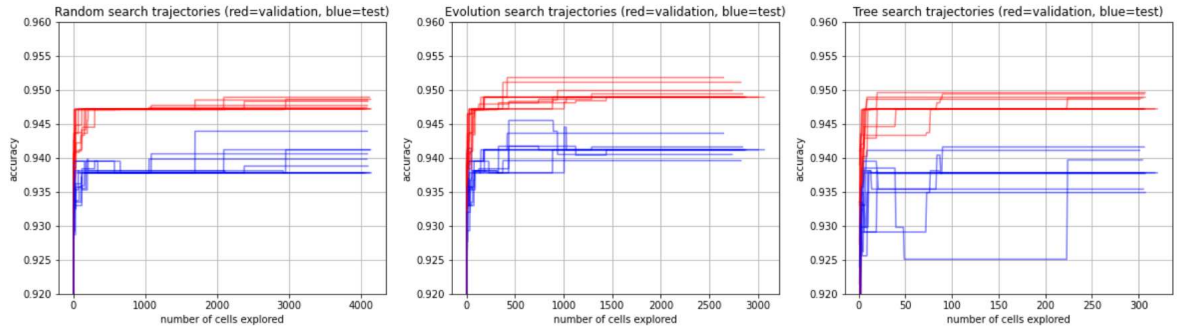

FIGURE 2: Comparison between different search algorithms by number of cells explored.

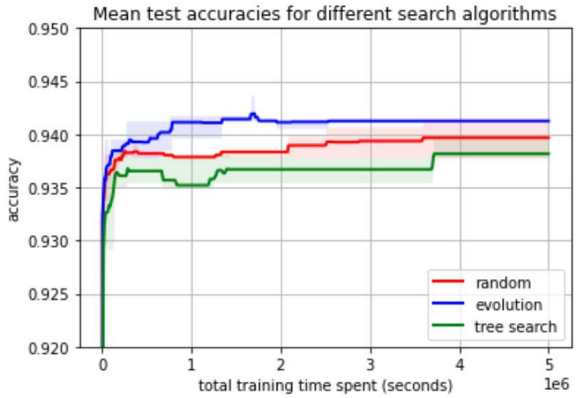

FIGURE 3: Comparison of mean accuracy achieved by different search algorithms.

3.2 Distance vs Accuracy

Since, tree search starts from a root node and small changes generate children, it depends on the initialization and subsequent children architectures would similar to those of their

ancestors. One possible reason that Tree Search underperforms compared to the other two algorithms might be that our data-set violates the assumption that "similar" nodes yield similar accuracy. So, we tested this assumption based on the methodology previously described. The result is shown in figure 4. As can be seen, with the pearson correlation of 0.11, there is no correlation between "distance" of nodes to each other and their accuracy.

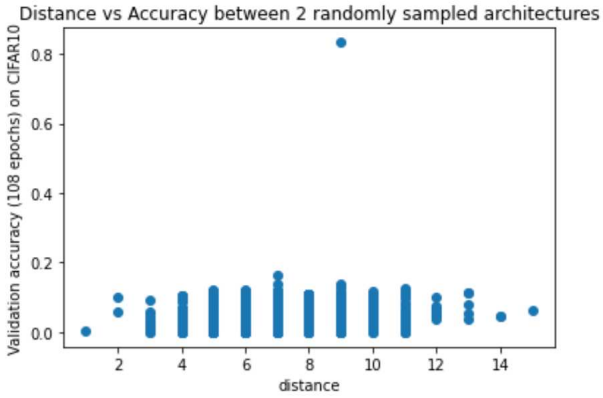

FIGURE 4

3.3 CIFAR-10 vs FashionMNIST Accuracy

We had a hypothesis that the CNN architecture which performs well on one dataset would also perform well on the other dataset. In figure 5, we show the test accuracy of randomly sampled architectures on both Fashion MNIST dataset and CIFAR-10 dataset. The correlation is very significant. This is a really astonishing result because these two datasets are very different.

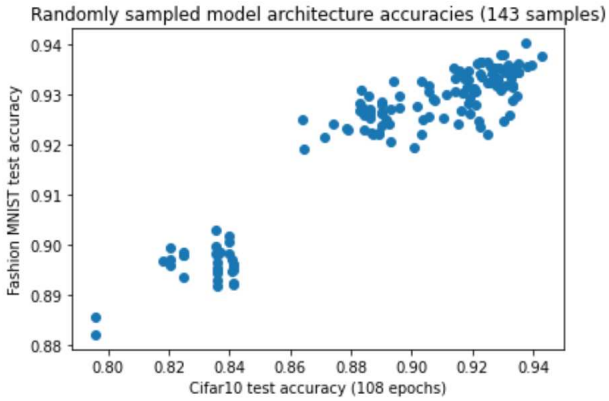

FIGURE 5

3.4 CIFAR-10 vs Imagenette Accuracies

To further confirm our hypothesis, we also tested the correlation between Cifar-10 and Imagenette datasets respectively. Imagenette dataset is a bit more complex with 3 channels

and higher resolution images. We could only train 21 CNNs for 36 epochs for the given time and resources. Still, to our astonishment, figure 6 shows that the accuracies are significantly correlated. Table 1 succinctly sums up the results of all the experiments we did. As can be seen, higher number of epochs yield higher correlation. It is because the higher the number of epochs, the more the CNN architecture matters.

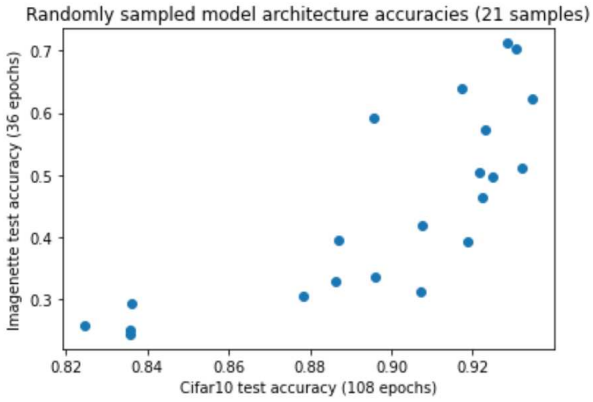

FIGURE 6

CIFAR-10	Fashion MNIST	Imagenette	p-val < 0.05
12 epochs	0.21	0.37	No
36 epochs	0.79	0.78	Yes
108 epochs	0.93	0.78	Yes

TABLE 1: Pearson Correlation Coefficient between accuracies of different datasets when trained on same CNN architectures.

3.5 Performant Cells

We randomly sampled top performant cells (accuracy > 0.92) and noticed some similarities in their architectures. For example, Input is always connected to Conv3X3 -> Batch Normalization -> ReLU. MaxPool usually occurs in the middle. The output could be connected to any of the possible layers. More investigation is needed to make statistically significant conclusions.

4 Conclusion

This was an attempt to get some insights into NASBench-101 dataset and to evaluate if it could be useful for our custom image classification problem. First, we discovered that the similar architectures don't yield similar accuracies. Although we did not discover a relationship between the architectures and the accuracies, we found that evolutionary search

algorithm does a great job in searching this architecture. Our experiments also show that the test accuracies of same CNN on different datasets are correlated. Thus, there may be some architectures which perform well despite of the dataset. This is a very useful insight for us as we can use NASBench-101 for our custom task. The future work may be to test this hypothesis on more datasets and extract the dataset invariant architectures of CNN.

References

[1] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck, *Monte-carlo tree search: A new framework for game ai.*, AIIDE, 2008.

[2] Aurélien Garivier and Eric Moulines, *On upper-confidence bound policies for non-stationary bandit problems*, arXiv preprint arXiv:0805.3415 (2008).

[3] Jeremy Howard, *Imagenette*, 2018.

[4] Alex Krizhevsky, Geoffrey Hinton, et al., *Learning multiple layers of features from tiny images*, (2009).

[5] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., *Mastering the game of go with deep neural networks and tree search*, nature **529** (2016), no. 7587, 484–489.

[6] Linnan Wang, Yiyang Zhao, Yuu Jinnai, Yuandong Tian, and Rodrigo Fonseca, *Alphax: exploring neural architectures with deep neural networks and monte carlo tree search*, arXiv preprint arXiv:1903.11059 (2019).

[7] Han Xiao, Kashif Rasul, and Roland Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, 2017.

[8] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter, *Nas-bench-101: Towards reproducible neural architecture search*, International Conference on Machine Learning, PMLR, 2019, pp. 7105–7114.

Appendix

4.1 Useful links

Project Git: https://github.com/akashjorss/NAS_Search_using_NASBench-101

Project Colab: https://drive.google.com/file/d/18xkCB5xDTW_XCvsAPvkbMSxmuBM9SNxH/view?usp

4.2 Contribution of Authors

4.2.1 Akash Malhotra:

1. Literature Review.
2. Evaluated search algorithms.
3. Developed the correlation hypotheses.
4. Worked on testing the correlation hypotheses.

4.2.2 Shubham Biswas:

1. Literature Review.
2. Collected datasets and worked on data-loader.
3. Contributed to the code-base.

4.3 NASBench-101 Architecture

NASBench is a tabular dataset which maps convolutional neural network architectures to their trained and evaluated performance on CIFAR-10. Specifically, all networks share the same network "skeleton", which can be seen in Figure (a) below. What changes between different models is the "module", which is a collection of neural network operations linked in an arbitrary graph-like structure.

Modules are represented by directed acyclic graphs with up to 9 vertices and 7 edges. The valid operations at each vertex are "3x3 convolution", "1x1 convolution", and "3x3 max-pooling". Figure (b) below shows an Inception-like cell within the dataset. Figure (c) shows a high-level overview of how the interior filter counts of each module are computed.

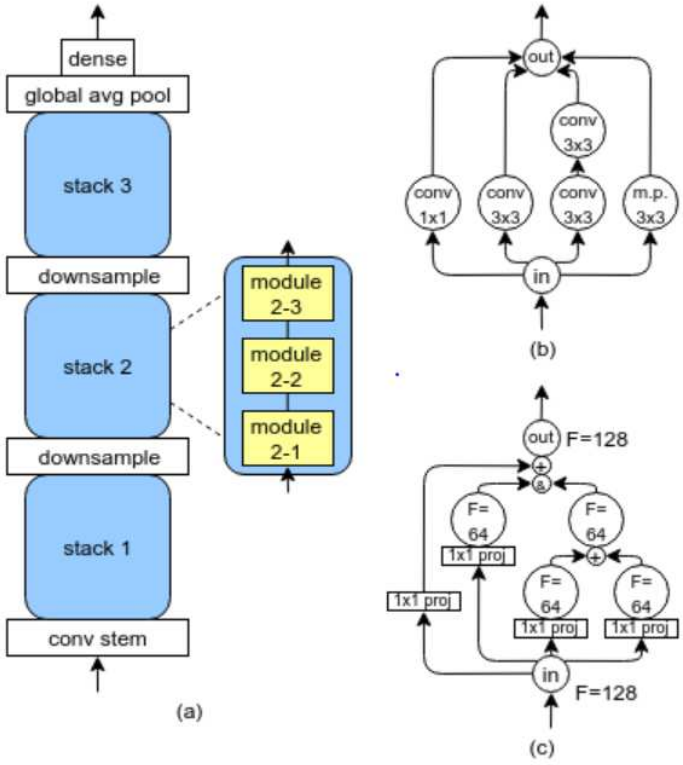

FIGURE 7: NAS Bench Architecture

4.4 AlphaX and Monte Carlo Tree Search

AlphaX explores the search space with a distributed Monte Carlo Tree Search (MCTS) [1] and a Meta-Deep Neural Network (DNN). MCTS guides transfer learning and intrinsically improves the search efficiency by dynamically balancing the exploration and exploitation at fine-grained states, while MetaDNN predicts the network accuracy to guide the search, and to provide an estimated reward to speed up the rollout.

The MCTS is to analyze the most promising move at a state, while the meta-DNN is to learn the sampled architecture performance and to generalize to unexplored architectures so that MCTS can simulate many rollouts with only an actual network training in evaluating a new node.

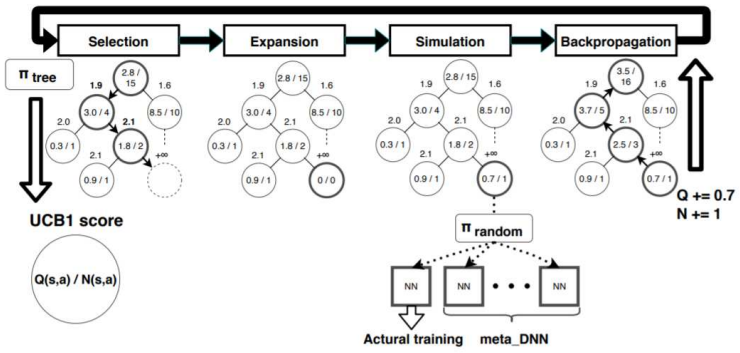

FIGURE 8: AlphaX Search Procedure