Algorithm for the best performing CNN architecture on the Fashion-MNIST data-set in NAS space.

Anukul Kumar
Roll No. B18ME013
IIT Jodhpur
Jodhpur, India
kumar.98@iitj.ac.in

Vinit Oswal
Roll No. B18ME040
IIT Jodhpur
Jodhpur, India
oswal.1@iitj.ac.in

Tarun Meena
Roll No. B18ME056
IIT Jodhpur
Jodhpur, India
meena.35@iitj.ac.in

***Abstract*—Advanced neural networks for specific tasks require weeks, if not months, of rigorous study by professionals with extensive subject knowledge.Many scholars have been interested in neural architecture search (NAS) in recent years because of its potential to automatically construct efficient neural architectures.Evolutionary algorithms have proven to be the most successful derivative-free optimization algorithms of all the search strategies.However there are many limitations due to the demand of high computational resource.To create best performing search model a test accuracy above 75% there need to be a trade-off between test accuracy and parameter count and constraints on the CNN architecture that your search function even though having very large space of all possible neural networks.**

I. Introduction

Although the majority of popular and successful model architectures are created by humans, this does not imply that we have thoroughly investigated the network architectural space and selected the optimal alternative[1]. If we use a systematic and automatic approach to learning high-performance model designs, we will have a better chance of finding the best option.Learning and developing network topologies automatically isn't a novel concept. In recent years, there has been a lot of interest in the topic of Neural Architecture Search (NAS), which has resulted in a lot of interesting ideas for better, quicker, and more cost-effective NAS approaches.

The goal of Neural Architecture Search (NAS) is to automate network architecture engineering such that it can learn the best network topology for a given task. Many new ideas for better, faster, and more cost-effective automatic neural architecture search emerged from splitting the approaches for NAS into three components: search space(describes a set of operations (for example, convolution, fully connected, and pooling) and how they can be coupled to construct viable network architectures.), search algorithm(population of network architecture candidates is sampled. It optimises to generate high-performance architecture candidates based on the child model performance parameters (e.g. high accuracy, low latency).), and child model evolution strategy(provide feedback for the search algorithm to learn, measure, estimate, or anticipate the performance of a large number of proposed child models).

The Evolutionary Algorithm is a type of algorithm that evolves over time. NEAT (NeuroEvolution of Augmenting Topologies) is a genetic algorithm-based method for evolving neural network topologies (GA). NEAT evolves both the link weights and the topology of the network at the same time. Each gene contains all of the data needed to configure a network, including node weights and edges. The population expands as a result of weight and connection mutations, as well as crossover between two parent genes.

II. Other related works

A. *Random Search*

The search space is directly determined by the neural architecture encoding method, but the neural architecture search algorithms are solely used to find the best neural architecture in the search space. Many previous knowledge may be incorporated to the neural architecture encoding ways to limit the search space and enhance the search time, such as skip-connection[3] can be beneficial in training deeper networks, etc. These human biases, on the other hand, hinder the ability of neural architecture search algorithms to discover unique and efficient structures.

III. Evolutionary Neural Architecture Search

Evolutionary algorithms, such as evolutionary programming (EP), evolutionary strategy (ES), and genetic algorithms (GAs), are a class of algorithms inspired by natural evolution and characterised by population-based and stochasticity [4]. Evolutionary algorithms, including mutation, crossover, and selection operators, may effectively solve optimization issues [5], even for black-box optimization problems in large-scale space. Since 20 years ago, there have been numerous research merging evolutionary algorithms with neural networks [6]. Evolutionary techniques have been extended to neural architecture search with the discovery of deep learning [12].The evolutionary neural architecture search can be divided into two groups in terms of individual representation: overall structural encoding and cell-based encoding. The former records the entire neural architecture as a single unit, whereas the latter simply encodes the cell, and the entire neural architecture is made up of a stack of identical cells.

IV. Solution Approach

Evolutionary algorithm has the following steps:
1. Random initialisation of some individuals (population)
2. Generating offsprings by combining two parent individuals
3. Mutating the generated offspring
4. Generating the model and finding the fitness accuracy of the child individuals
5. Adding the children individuals to the main population.
6. Removing the worst performing individuals from the main population space.

These steps are followed for a certain number of epochs. At the end of the process, only the best performing individuals remain in the main population space. The algorithm reaches a steady after a certain number of epochs.

In the final step, the best performing individual from the final main population space is selected and returned as output by the algorithm.

V. Implementation of the algorithm in Python and Google Colab

The functions used in the algorithm are as follows:

1.genome-maker (genome, layers, reduction-layers)-

In a case where the resultant genome string obtained after combining two parent genome results in a genome with less that 2 rc layers , this function is called.The function generates 1 or 2 reduction layers depending on the requirement and adds these layers to the original genome string to produce a result which can be parsed in the train model function

2.mutate (genome)-

This function is used to perform mutations om the genome string generated by mating two parent genomes and generate a child function to be trained. The mutation algorithm changes the values of the parameters namely kernel, number of CNN filters of all the layers of the input genome string and returns the resultant string as output

3.convert-to-genome (nc-parameters, rc-parameters, rc-locations, fl-active)-

In this function the random values generated for the layers of a genome are converted into a string format which can be parsed as an arguement to the train-model function.

4.make-children (p1, p2, alpha)-

This function is used to generate children genomes for the selected parent genomes. For the case of this problem, we take two parents to generate the child genome. The indexes of the parent genomes are generated randomly and are chosen from the main all-genomes. We also give one more argument i.e alpha. This parameter helps us determine the point where the parent strings are broken and are added to get the final genome. In this problem, we take alpha as 0.5. This means that the parents strings are broken into half and these half parts are connected to get the final genome string.

The final genome string is then checked to see if all the constraints of a genome string are met. If they are not met, it is processed further to get a string that can be processed by the train-model function.

After this step, the mutate function is called and the resultant mutated string is returned to be sent to the CNN model for training.

5.random-init-genome (max-nc)-

This function is used to randomly initialise the parameters of the layers.

VI. Results and Conclusions

The given problem statement was to develop an algorithm to search in the NAS space for the best performing neural network architecture for the fashion MNIST dataset. We are given some constraints for the network and it's normal as well as reduction convolution layers. We are also given the structure of the final layers. We have to find the structure that follows these constraints and also gives the best performance.

In this project, we have used an evolutionary search algorithm for this purpose. We build a evolutionary search algorithm from scratch and decide the hyperparameters of the model by hit and trial. We then implement the algorithm using a individual space of 20 cases and pair two parents to generate children from them. We generate 5 children from 10 randomly chosen pairs of parents. After mutation, the best performing genome models are stored and others are discarded.
After 50 epochs, we reach to the following genome string:

NC 42 2 gelu;NC 82 5 tanh;NC 14 2 tanh;RC 31 5 swish;NC 52 2 tanh;RC 86 3 relu;NC 35 7 relu;FL swish;

The test case accuracy of this genome string is : 91.51999950408936 %

Hence, the evolutionary search algorithm is successfully implemented to determine the network architecture for a CNN network based on the Fashion-MNIST dataset

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no. 7553, pp. 436–444, 2015.

[2] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," arXiv preprint arXiv:1611.02167, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016

[4] T. Back, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press, 1996.

[5] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," IEEE Transactions on Evolutionary computation, vol. 3, no. 2, pp. 82–102, 1999

[6] X. Yao, "Evolving artificial neural networks," Proceedings of the IEEE, vol. 87, no. 9, pp. 1423–1447, 1999

[7] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," arXiv preprint arXiv:1611.01578, 2016