CNN Architechture Search

Deepak Arjariya
B18ME021
IIT Jodhpur
India
arjariya.1@iitj.ac.in

Ashish Ledalla
B18EE008
IIT Jodhpur
India
ledalla.1@iitj.ac.in

Hritu Raj
B18ME023
IIT Jodhpur
India
raj.9@iit.ac.in

***Abstract*—The Neural Architectural Search methodology is an automated architecture search methodology for neural networks . Convolutional Neural Networks (CNNs) created using NAS approaches have recently produced outstanding results in a variety of applications, including image classification and natural language processing. This project is in the same NAS domain, and we use an evolutionary algorithm augmented with a multi-objective non-dominated sort algorithm (NSGA II) [2] to traverse the search space of neural network topologies.**

I. Introduction

In this course project, we have to create a search algorithm to search in Neural architechture search space for the best performing Convolutional Neural Network(CNN) architechture on the fashion-mnist data-set. Best performing is defined as having a test accuracy above 75 percent and the lowest parameter count model the algorithm can find.There is trade-off between test accuracy and parameter count and the algorithm tries to balance the trade-off. This problem is treated as a bi-objective non-dominated sorting evolutionary system and using NSGA IIs [2] and a custom designed algorithm designed search for the best architecture in the search space.

II. Search Strategy

We have used evolutionary alghorithm in which our population tries to better itself after each generation. First we randomly initialized all the architechture models. Among all the models we have taken the best candidates using NSAG-2 alghorithm and have selection rate of 0.6. After that as we do not reject all the models which are not elite. Among all the rejected models we have selected random models with 1- selection rate. We put all the randomly selected model and best selected model in the current population. Now, to improve the models in current population we mate certain models depending on mate rate(which is chosen to be 0.6), We also mutate the random sample of population, and add all of them to the current population and then iterate for certain number of generations. The Architecture model is divided into different layers and each layer is represented as a list which has 4 elements except the last layer, 1st element represent if it is RC or NC which is denoted by 0 and 1 respectively, second element represents the number of filters, third element represents kernel size and the last element represents activation function denoted by integer from 0-4 for each in [0 - relu, 1 - sigmoid, 2 - tanh, 3 - swish, 4 - gelu]

A. Algorithm

Algorithm 1 Evolutionary algorithm

Evolutionary Inputs: N_g, R_{mu},R_{mix},S_r
Training Inputs:lr=1e-3, $lr_decay = 0.1$, $epoch = 1$

$Po \Longleftarrow initializepopulation(Np)$
for(i $\Longleftarrow$ 0.....N_g)do
  $P_g \Longleftarrow PartialTraining(P_i, T_{param})$
  $E_v \Longleftarrow EvaluateAccuracyAndParameters(Pi)$
  $P_{best} \Longleftarrow BestSelection(P_i, E_v)$
  $P_r \Longleftarrow randomSelection(P_i)$
  $update\ P_i \Longleftarrow P_{best} + P_r$
  $P_{rc} \Longleftarrow MateCandidates(P_i, R_{mix})$
  $P_{mu} \Longleftarrow (P_i, R_{mu})$
  $updateP_i \Longleftarrow P_{mu} + P_{rc} + P_{remaining}$
end
$E_v \Longleftarrow EvaluateAccuracyAndParameters(P_{N_g})$
$returnBestCandidate(E_v)$

In the above algorithm, N_g represent number of generations, R_{mu} represent the mutation rate, R_{mix} represent the mixing rate and S_r represent selection rate. Based on the population size N_p, we initialize random architectures and then consider them for mating based on their performance according to the heuristic.

We used Partial Training for improving the performance of the algorithm, instead of training all the models in the population for 10 epochs every time,we decided we would run the population for 1 epoch to get the general idea about the accuracy and number of parameters. Then after we found the best models we then ran each model to 10 epochs to find the best among them.

III. heuristics

A. Naive Objective Algorithm

We designed this algorithm for selecting best candidate based on two parameters as shown in the algorithm above. This algorithm takes into consideration that we need to get maximum accuracy while also keeping the parameters count low. So we designed a cost function which gives us cost value based on accuracy and parameters. The cost value is

W_{ac}*Acc+W_{par}*(10000/N_p), where W_{ac} is weight of accuracy which is equals to (1-8/15*Acc) and W_{par} is the weight of the parameter which is equals to 1-W_{ac}, and Acc is Accuracy of the model which takes into consideration both test and train accuracy which equals to 0.8*testAcc+0.2*trainAcc. If the accuracy value is high the weight of accuracy will be low and vice versa. since we want to converge at higher accuracy than 0.75, so we kept the value of W_{acc} as 0.60 at 0.75 accuracy.

B. NSGA-2 Algorithm

NSGA-2 is a fast nondominated sorting approach that utilizes Pareto front and the maintains diversity through crowd distancing. The algorithm is specifically for optimization problems with multiple objectives. In this case of NAS, the objectives considered are test accuracy and number of parameters of the respective model. since number of parameters is always a positive quantity, we simply take the reciprocal to have two maximization functions. Using NSGA 2, the population is sorted after their evaluation. In the algorithm shown above, the BestSelection is used as a general term for the heuristc.

IV. Results

A. Using Naive Objective Algorithm

Gnome string found – "NC 52 5 swish;NC 80 3 swish;NC 37 7 swish;NC 48 5 relu;RC 63 2 tanh;NC 101 4 swish;NC 57 1 swish;RC 79 1 relu;FL swish;"

The following results are for 10 epochs.

testAccuracy – 0.8906000256538391
trainAccuracy – 0.9013500213623047
nParameters – 358631

Replacing tanh and sigmoid with relu in the above genome string slightly improved the results as follows, all the results are 10 epochs

testAccuracy – 0.894599974155426
trainAccuracy – 0.9121500253677368
nParameters – 358631

B. Using NSGA2 Algorithm Algorithm

Gnome string found – "NC 67 4 relu;RC 82 3 tanh;NC 40 3 sigmoid;NC 94 3 swish;NC 68 4 sigmoid;NC 61 5 relu;RC 82 6 relu;FL gelu;"

testAccuracy – 0.9017999768257141
trainAccuracy – 0.9132500290870667
nParameters – 506378

Replacing tanh and sigmoid with relu in the above genome string slightly improved the results as follows, all the results are 10 epochs

testAccuracy – 0.911899983882904
trainAccuracy – 0.9278166890144348
nParameters – 506378

References

[1] Sapra, Dolly, and Andy D. Pimentel. "Designing convolutional neural networks with constrained evolutionary piecemeal training." Applied Intelligence (2021): 1-15.

[2] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE transactions on evolutionary computation 6.2 (2002): 182-197.