ABSTRACT

We propose Efficient Neural Architecture Search (ENAS), for automatic model design which is fast and efficient. Here, a controller is trained to build child network with micro search strategy. It is trained to maximize the expected validation accuracy of child network while minimizing cross entropy loss. This is performed via parameter sharing with child models on MNIST and CIFAR-10 datasets. This method is faster than other existing models and gives commendable results using GPU hours. It is motivated by the reinforcement learning to build CNNs and RNNs. NAS has a drawback where the model parameters are optimized from scratch each time. The discovered architecture achieved test accuracy of 99.77% on MNIST dataset.

Chapter No.	TABLE OF CONTENTS	Page No.

1.	**INTRODUCTION**	**4**
	1.1 Preamble	**4**
	1.2 Motivation	**4**
	1.3 Objectives of the project	**4**
	1.4 Literature Survey	**5**
	1.5 Problem Definition	**6**
	1.6 Problem Statement	**6**
2.	**PROPOSED SYSTEM**	**7**
	2.1 Description of Proposed System.	**7**
	2.2 Description of Target Users	**7**
	2.3 Advantages of Proposed System	**8**
	2.4 Scope	**8**
3.	**SOFTWARE REQUIREMENT SPECIFICATION**	**9**
	3.1 Overview of SRS	**9**
	3.2 Requirement Specifications	**10**
	3.2.1 Functional Requirements	**10**
	3.2.2 Non-Functional Requirements	**10**
	3.3 Software and Hardware requirement specifications	**10**
4	**SYSTEM DESIGN**	**11**
	4.1 High Level Design	**11**
	4.2 System Architecture	**12**
	4.3 Flow Chart	**12**
5	**IMPLEMENTATION**	**13**
	5.1 Proposed Methodology	**13**
	5.2 Description of Mode	**14**
6	**RESULTS AND DISCUSSIONS**	**15**
7	**CONCLUSION AND FUTURE SCOPE**	**18**
8	**REFERENCES**	**19**

Chapter 1

Introduction

1.1 Preamble

- Developing neural network models often requires significant architecture engineering.
- It’s a lot of trial and error and the experimentation itself is time consuming and expensive.
- Our aim is to automate the process of architecture engineering using Neural Architecture Search (NAS).

1.2 Motivation

- The process of designing neural network is incredibly resource consuming and it may not be efficient.
- It is challenging to find architecture that satisfies accuracy, memory and latency constraints.
- NAS is one of the quick way of getting great accuracy for machine learning task without much work.

1.3 Objectives of the Project

- To find suitable neural network architecture for given problem and dataset with less GPU days.
- The architecture given by NAS should perform best among all other architecture for that given task when trained by the dataset provided.

1.4 Literature Survey

1.4.1 Title: Neural architecture search (NAS) - The Future of Deep Learning (2019)

This article gives brief introduction to NAS, three different NAS methods, dimensions of NAS method. Search strategy and performance evaluation is explained with three different NAS methods.

1.4.2 Title: Project Petridish: Efficient Forward Neural Architecture Search (2019).

In this article the importance of neural architecture search, two different types of neural architecture search is discussed.The article explains about petri dish which is a NAS algorithm to optimize the selection of neural network and three different phases in petri dish

1.4.3 Title: Neural Architecture Search with Reinforcement Learning (2016)

In this paper recurrent neural network is used to generate the model descriptions of neural networks and the RNN is trained with reinforcement learning to maximize the expected accuracy of the generated architectures on a validation set. On CIFAR-10 model achieves a test error rate of 3.65, which is 0.09 percent better and 1.05x faster than the previous state-of-the-art model that used a similar architectural scheme.

1.4.4 Title: Title: Efficient Forward Architecture Search (2019)

In this paper Petri Dish algorithm is proposed. The proposed algorithm is motivated by the feature selection algorithm forward stage-wise linear regression. In order to reduce the number of trials of possible connection combinations, they have jointly trained all possible connections at each stage of growth while leveraging feature selection techniques to choose a subset.

1.5 Problem Definition

For any machine learning problem the aim is to determine the best neural network architecture for that particular problem. The process of designing neural network is time consuming and the neural network designed may not be efficient.

Our aim is to create a deep neural network which will decide the best architecture for the given dataset.

1.6 Problem Statement

Finding Optimal Neural Network Architecture for a Machine Learning Problem Using Neural Architecture Search.

Chapter 2

Proposed System

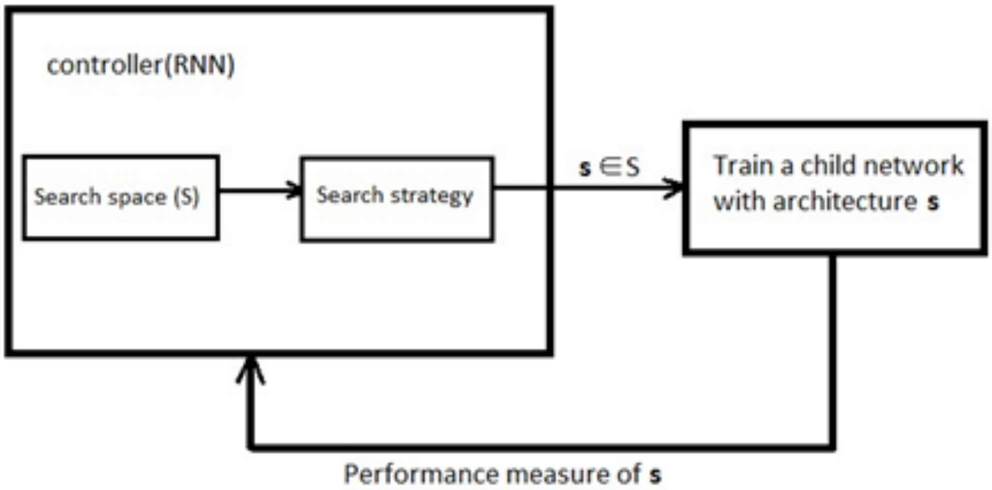


Figure 2.1: Proposed System

2.1 Description of proposed system

The Controller (RNN) controls the building of the child model's architecture using search strategy. A generated child model is one of the many possible child models that can be built in the **search space (S)** .This particular child model is trained to convergence using stochastic gradient descent to minimize the expected loss function between the predicted class and ground truth class. The controller's parameters are updated, to maximize the expected reward function which is the validation accuracy.

2.2 Description of Target Users

Data Scientists are the Target Users.

Data Scientists will be able to obtain a neural network architecture that satisfies not only accuracy but also memory and latency constraints for any machine learning problem quickly.

2.3 Advantages of proposed systems

- The manual process of designing neural network architecture is automated.
- The best architecture for any machine learning problem can be obtained in terms of memory, latency and accuracy.
- We can obtain great accuracy for machine learning task without much work.

2.4 Scope

- It becomes easy to find an architecture with reasonable performance for machine learning task.

- The manual process of designing neural network is automated.

Chapter 3

Software Requirements Specifications

Software Requirement Specification explains about necessary hardware and software components needed for the smooth working of the system, some of the software and hardware requirements are listed along with the functional and non-functional requirement of the system. Functional requirements provide us with the list of functionalities to be adopted by the system. Non-functional requirements are the requirements that specify the performance parameters of the system.

3.1 Overview of Software Requirement Specifications:

The introduction of the software requirement specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, reference and overview of the SRS.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

3.2 Requirement Specifications

3.2.1 Functional Requirements

- The architecture shall be optimal for the given dataset.
- The architecture given by NAS shall perform best among all other architecture for that given task when trained by the dataset provided.

3.2.2 Non-Functional Requirements:

- NAS should take less GPU days.
- Evaluation of Neural Network Architecture should be faster

3.3 Software and Hardware Requirement Specifications:

3.3.1 Software Specification:

- OS: Windows 10(Ubuntu 16.04)
- Open CV 3.4.1
- Python 3.5

3.3.2Hardware Requirements:

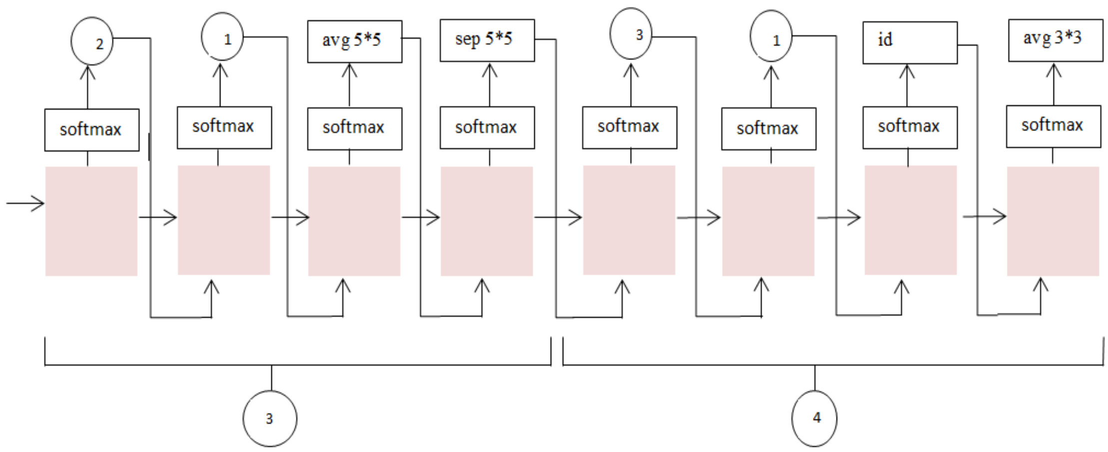
- Graphic Card/RAM:1080TI/32GB

Chapter 4

System Design

4.1 High Level Design

Controller



Child Model

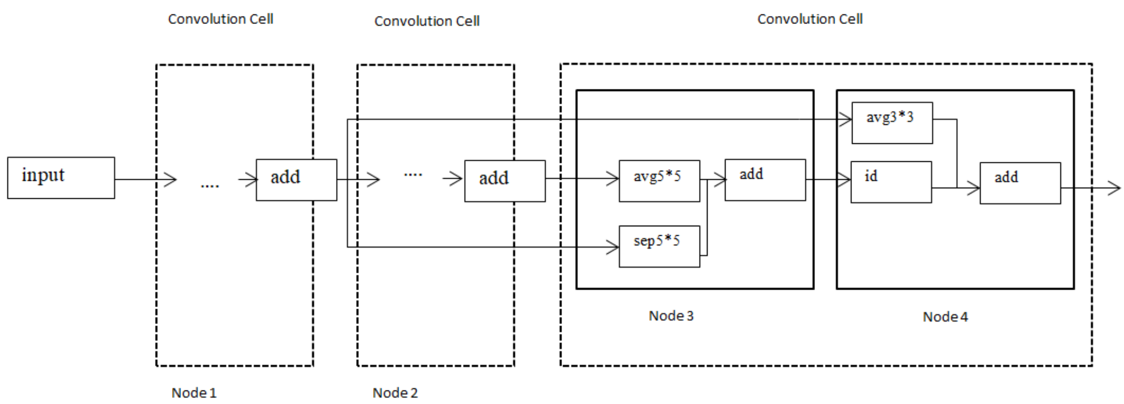


Figure 4.1:High Level Design

The proposed system uses micro search strategy for generating neural network architecture. Micro search designs only one building block whose architecture is repeated throughout the final architecture. These building blocks are convolutional cell and reduction cell.

4.2 System Architecture

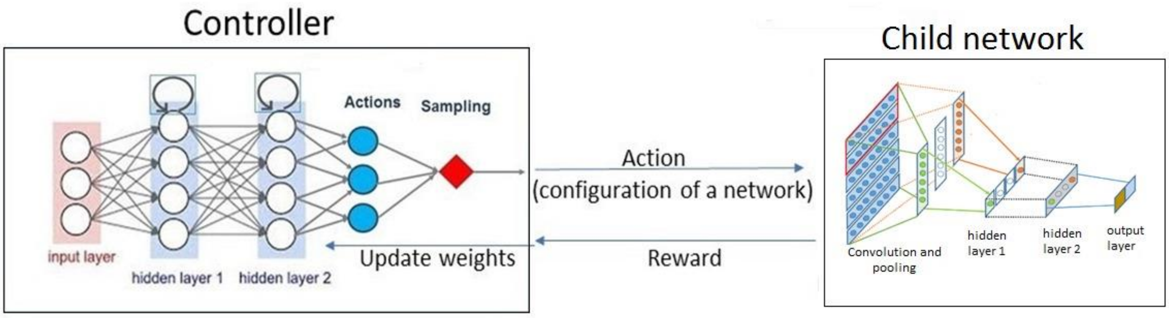


Figure 4.2:System Architecture

4.3 Flow chart

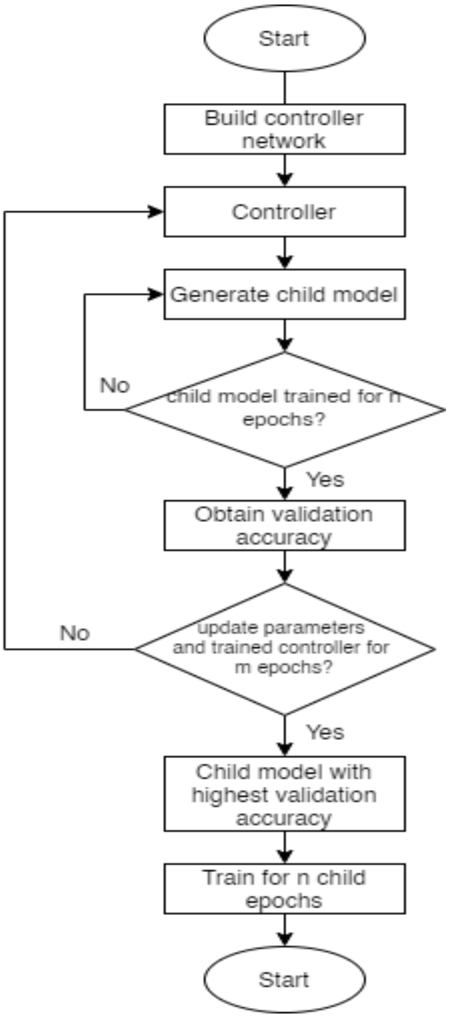


Figure 4.3:Flow Chart

Chapter 5

Implementation

This chapter contains the detailed technical description of the proposed methodology along with description of various modules used in the system and snapshots of the respective code and their explanations.

5.1 Proposed methodology:

ENAS has 2 types of neural networks controller and child model. Certain number of epochs are set for controller and child model and control network is built. In each epoch, a child model is generated from search space using directed acyclic graph(DAG) and trained for certain number of epochs for which validation accuracy is calculated. The controller parameters are updated and the whole process is repeated for all the epochs. After the controller is trained, the child model with the highest validation accuracy is obtained and trained for same number of child epochs. For our model MNIST dataset gave commendable result with controller batch size of 160 for 150 epochs and for child model with batch size of 128 for 309 epochs. The search strategy used by the controller to generate child model is micro search which designs a single building block as convolution and reduction cell. Child model consists of several nodes. The hyperparameters like the number of nodes and convolutions cells are tuned to get better accuracy. In the final child model, all the convolution cells share th3 same architecture.

Algorithm

```
CONTROLLER_EPOCHS = X
CHILD_EPOCHS = Y
Build controller network
for i in CONTROLLER_EPOCHS:
        Generate a child model
        Train this child model for CHILD_EPOCHS
        Obtain val_acc
        Update controller parameters
Get child model with the highest val_acc
Train this child model for CHILD_EPOCHS
```

5.2 Description of Model:

Controller is an LSTM with 100 hidden units trained using adam optimizer. Child model is the desired CNN for image classification. The controller generates the child model architecture using search strategy using certain operations like convolution, pooling etc. Search space has exponential number of configurations. In experiment if 4 activation functions are used with N nodes then search space has (4^N) x N!. The child model is trained to convergence in order to minimize the loss and maximize the validation accuracy. The two sets of learnable parameters are LSTM(θ) of controller and (ω)shared parameter of child model. Controller's parameters is updated using REINFORCE to maximize the expected reward. It is trained by reinforcement learning where controller acts as agent with an action of decision taken to build child network while maximizing the reward i,e. validation accuracy of child network. The shared parameters of child models are trained using SGD.

We trained our model with MNIST handwritten digit dataset with 55000 training data, 10000 test data and 5000 validation data.

ENAS includes 3 concept of search space, search strategy and performance evaluation. Search space is a set of all possible architectures that can be generated. Search strategy is a method to generate child networks and performance evaluation is a method to measure the effectiveness of the generated child models.

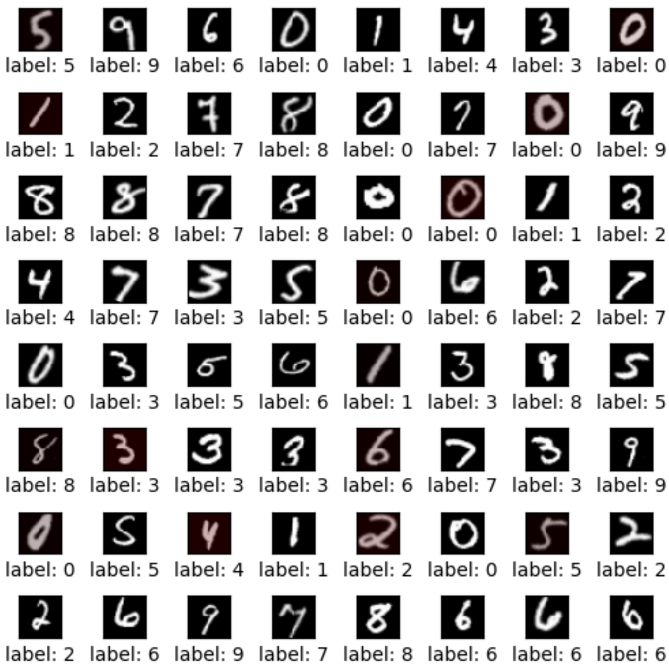


Figure 5.1 : MNIST Augmentation

Chapter 6

Results and Discussions

Controller's Training output

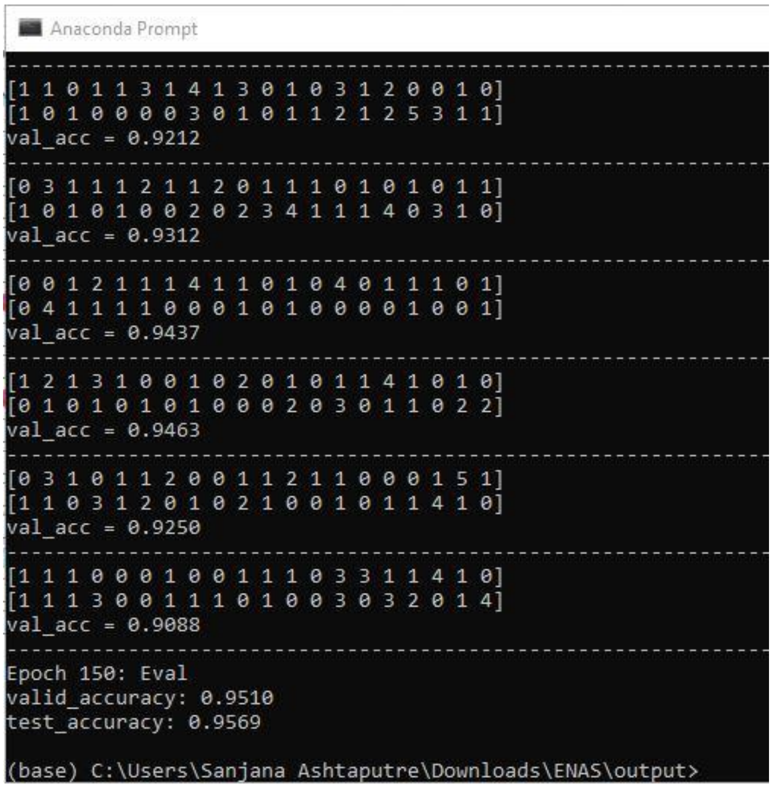


Figure 6.1 : Controller's Training Output

Total epochs: 150

Batch size: 160

Validation accuracy: 0.9510

Test accuracy: 0.9569

Child model architecture:

"1 2 1 3 0 1 0 4 1 1 1 1 0 1 0 1 1 0 0 1

(architecture for convolution layers)

0 1 0 4 1 0 2 0 0 3 1 1 0 0 0 0 4 1 1 0"

(architecture for pooling layers)

Final child model training Output

```
Anaconda Prompt
epoch = 308   ch_step = 397500 loss = 0.001094   lr = 0.0000   |g| = 0.2149   tr_acc = 128/128   mins = 2176.66
epoch = 308   ch_step = 397550 loss = 0.002354   lr = 0.0000   |g| = 0.0946   tr_acc = 128/128   mins = 2176.93
epoch = 308   ch_step = 397600 loss = 0.003172   lr = 0.0000   |g| = 0.3729   tr_acc = 128/128   mins = 2177.20
epoch = 308   ch_step = 397650 loss = 0.001986   lr = 0.0000   |g| = 0.0697   tr_acc = 128/128   mins = 2177.46
epoch = 308   ch_step = 397700 loss = 0.001657   lr = 0.0000   |g| = 0.1831   tr_acc = 128/128   mins = 2177.72
epoch = 308   ch_step = 397750 loss = 0.001570   lr = 0.0000   |g| = 0.0452   tr_acc = 128/128   mins = 2177.99
epoch = 308   ch_step = 397800 loss = 0.002609   lr = 0.0000   |g| = 0.1151   tr_acc = 128/128   mins = 2178.25
epoch = 308   ch_step = 397850 loss = 0.005959   lr = 0.0000   |g| = 4.1540   tr_acc = 128/128   mins = 2178.52
epoch = 308   ch_step = 397900 loss = 0.003682   lr = 0.0000   |g| = 0.6595   tr_acc = 128/128   mins = 2178.78
epoch = 308   ch_step = 397950 loss = 0.001732   lr = 0.0000   |g| = 0.0639   tr_acc = 128/128   mins = 2179.05
epoch = 308   ch_step = 398000 loss = 0.001789   lr = 0.0000   |g| = 0.0674   tr_acc = 128/128   mins = 2179.31
epoch = 308   ch_step = 398050 loss = 0.001846   lr = 0.0000   |g| = 0.0565   tr_acc = 128/128   mins = 2179.58
epoch = 308   ch_step = 398100 loss = 0.000945   lr = 0.0000   |g| = 0.0569   tr_acc = 128/128   mins = 2179.84
epoch = 308   ch_step = 398150 loss = 0.000965   lr = 0.0000   |g| = 0.0388   tr_acc = 128/128   mins = 2180.11
epoch = 308   ch_step = 398200 loss = 0.001164   lr = 0.0000   |g| = 0.0615   tr_acc = 128/128   mins = 2180.37
epoch = 308   ch_step = 398250 loss = 0.001702   lr = 0.0000   |g| = 0.2606   tr_acc = 128/128   mins = 2180.64
epoch = 308   ch_step = 398300 loss = 0.002423   lr = 0.0000   |g| = 0.0510   tr_acc = 128/128   mins = 2180.90
epoch = 308   ch_step = 398350 loss = 0.002024   lr = 0.0000   |g| = 0.1681   tr_acc = 128/128   mins = 2181.17
epoch = 308   ch_step = 398400 loss = 0.001829   lr = 0.0000   |g| = 0.1139   tr_acc = 128/128   mins = 2181.43
epoch = 308   ch_step = 398450 loss = 0.002104   lr = 0.0000   |g| = 0.1022   tr_acc = 128/128   mins = 2181.70
epoch = 308   ch_step = 398500 loss = 0.001744   lr = 0.0000   |g| = 0.1451   tr_acc = 128/128   mins = 2181.96
epoch = 308   ch_step = 398550 loss = 0.002017   lr = 0.0000   |g| = 0.1715   tr_acc = 128/128   mins = 2182.22
epoch = 308   ch_step = 398600 loss = 0.001198   lr = 0.0000   |g| = 0.0704   tr_acc = 128/128   mins = 2182.49
Epoch 309: Eval
Eval at 398610
test_accuracy: 0.9977
('n_Error is: 0.0023)
('true label is: ', [8, 9, 9, 6, 8, 6, 4, 5, 5, 5, 7, 2, 9, 6, 7, 6, 9, 7, 3, 4, 1, 7, 6])

(base) C:\Users\Sanjana Ashtaputre\Downloads\ENAS\output>
```

Figure 6.2 : Child model training output

Total epochs: 309

Batch size: 128

Test accuracy: 0.9977

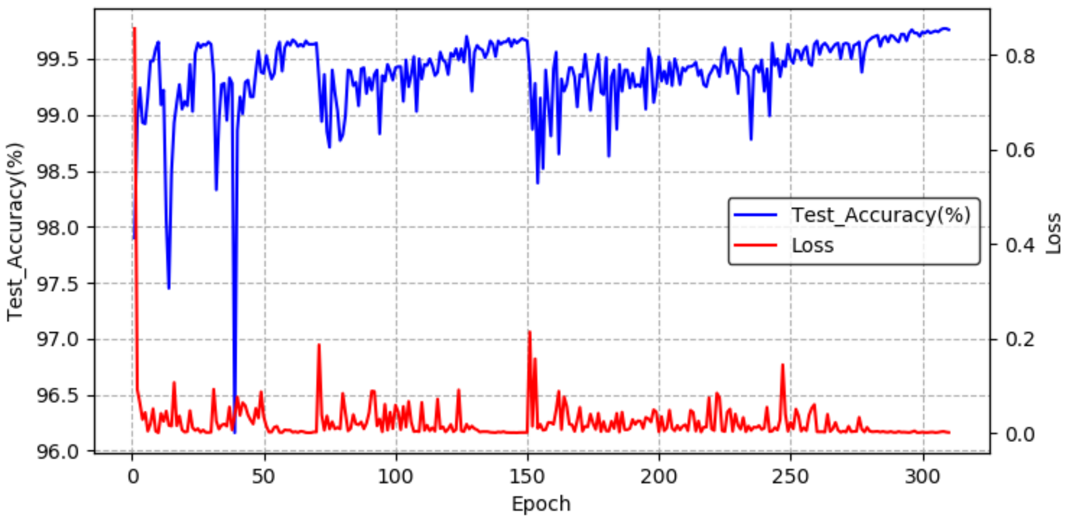


Figure 6.3 : Child Network Loss & Test Accuracy

Comparative Analysis

Table 6.1 : Comparative analysis with the existing models

Method	**Test accuracy**(%)	**Error**(%)	**Parameters** (million)	**GPUs**	**Days**
ENAS + micro search Space	99.77	0.23	4.6	1	0.52
NAS	95.53	4.47	7.1	800	21-28
DenseNet	96.54	3.46	25.6	-	-
ENAS + macro search space	95.77	4.23	21.3	1	0.48

From the above table, we can state that our modes i,e. ENAS with micro search space proves to be one of the best models in terms of accuracy, parameters, number of GPUs and the total time required to train the model. When compared to NAS, it shows a drastically improved results mainly in terms of number of GPUs and time taken to train the model. As observed in the above table, the total number of parameters is lesser compared to the other existing models. In ENAS with macro search space the whole model is built instead of blocks. In terms of number of days and number of GPUs required for training it is on par with our model ENAS with micro search but a setback with macro search is the total number of parameters and certainly the error rate. DenseNet i,e. Densely connected convolutional networks which is an extension of ResNet say with some fundamental difference. It is considered better for its flow of information and gradients in the network making it easy to train. On comparision with DenseNet too our model worked better in all the parameters. Perticularly in terms of number of parameters which is way more less than DenseNet parameters. In conclusion, ENAS with micro search space gave better results compared to other existing models.

Chapter 7

Conclusion and Future Scope

NAS is used to automate the designing process of the neural networks. But due to its computational expenses and drawbacks it restricts much usage in bigger projects. It uses around 400 GPUs and takes minimum 3 days for the training. The model parameters in NAS need to be optimized from scratch each time which is a major setback. To overcome these issues, ENAS is built which takes less than a day on one GPU for its training. It does not require the optimization of its parameters from scratch. Parameter sharing plays a major role in overcoming the NAS's setback. On comparing with other existing models, it proved to be one of the best in terms of many parameters.

We experimented the model on MNIST dataset which gave a very good result. In future the same can be applied to any other dataset to obtain better results than the existing models and the process to be automated rather than manually designing the networks.

8. References

[1] Md Ashiqur Rahman, “Neural architecture search (NAS)- the future of deep learning”, 9 June 2019

[2] Jesus Rodriguez, “Project Petridish: efficient forward neural architecture search”,13 January 2020, Microsoft.

[3] Barret Zoph, Quoc V. Le, “Neural architecture search with reinforcement learning”, 2017, Google Brain.

[4] Hanzhang Hu, John Langford, Rich Caruana,Saurajit Mukherjee, Eric Horvitz, Debadeepta Dey, “Efficient Forward Architecture Search”, 31 May 2019, Microsoft Research.

[5] Hieu Pham Melody Y. Guan , Barret Zoph, Quoc V. Le, Jeff Dean , “ Efficient Neural Architecture Search via Parameter Sharing”, 12 February 2018

[6] Raimi Karim, “ Illustrated: Efficient Neural Architecture Search”, 26 March 2019