Optimising Neural Network Architecture and Weights using Evolutionary Algorithms

Avish Kadakia
Student Id:1132452
Lakehead University
Thunder Bay, Ontario, Canada
email: kadakiaa@lakeheadu.ca

Manpreet Bhatti
Student Id: 1142587
Lakehead University
Thunder Bay, Ontario, Canada
email: bhattim@lakeheadu.ca

Supervised By:
Dr. Jinan Fiaidhi
Lakehead University
Thunder Bay, Ontario, Canada

***Abstract*—TThe 21st century has proven to be a pioneering era and a cornerstone for a plethora of innovations in AI (Artificial Intelligence)almost regularly. In a discipline that is so highly contested and always has some cutting-edge technologies to be discovered by researchers at every junction, We have decided to take a few steps back and try to optimize one of the fundamental concepts of AI training networks. Researchers have been attempting to tackle the same problem by using several techniques such as Knowledge Distillation, Neural networking training optimization using EA, Weight Agnostic Neural Network. We will not focus on creating something groundbreaking but would focus on gaining a deeper understanding of previous implementations and attempting to either modify or optimize them to yield even better results.**

I. INTRODUCTION

Neural networks have formed the basis for the current revolution in machine learning. The idea of creating neural networks originated from the thought of replicating how a human brain functions. There are various types of neural networks present today, but they can be boiled down in their simplest form as many interconnected nodes form several layers. Just the way neurons of a human brain are interconnected.

Backpropagation is one of the most popular techniques currently used to optimize the weights of a neural network in most cases. But there are several issues in using this technique, such as the network getting stuck in local optimum or not adapting to changes in the number of nodes or layers. This research will demonstrate whether it would be possible to use evolutionary algorithms to optimize the training for these neural networks and achieve better results. The primary motivation behind this research is that there are many fundamental concepts in AI that use machine learning and, more importantly, neural networks as their base. If I can optimize the most basic building block, it will automatically improvise all other associated research areas such as RNNs, Reinforcement Learning, CNNs, etc.

II. RELATED WORK

Machine learning is a vast field, and there are many areas of optimization. We believe that if we are to replicate or even surpass human intelligence using machine learning and neural networks, we need to use alternatives apart from backpropagation. We need to incorporate evolution. Because as we can see in the real world, development has played a significant role in shaping our intelligence. Multiple research papers propose several methodologies to achieve this goal. Listed below are a few of the techniques that are implemented to achieve the desired results

A. Knowledge Distillation

Their main idea is to use a larger (Teacher) neural network to train other smaller (Student) neural networks. As larger models will generalize better, and smaller models are best suited for specialization problems. Let us imagine a complicated task, like image classification for thousands of classes. Often, you can't just use an enormous pre-trained neural network sort of a ResNet50 and expect it to attain 99% accuracy. Therefore one can build an ensemble of models, equalizing the flaws of each other. However, this ensemble of models cannot be used for real-time applications because of the extensive time complexities. However, the model generalizes pretty well to the unseen knowledge. Therefore it's safe totrust its predictions. What if we use the predictions from the big and cumbersome model to train a smaller, supposed student model to approximate the larger one? This is knowledge distillation in essence, which was introduced within the paper Distilling the Knowledge in a Neural Network [1]. In broad strokes, the method followed is listed below:

- Train a large model that performs and generalizes very-well. This is called the teacher model.
- Take all the data you have, and compute the predictions of the teacher model. The total dataset with these predictions is called the knowledge, and the predictions themselves are often referred to as soft targets. This is the knowledge distillation step
- Use the previously obtained knowledge to train the smaller network, called the student model

B. Evolutionary algorithm neural network training optimisation

The basic idea proposed by the paper [2] to train neural networks is as follows:

- Create a population of several neural networks.
- Assign random weights or parameters to all the neural networks.

Perform the following for some amount of iterations:

- We train all the neural networks simultaneously side by side.
- Calculate the training cost after they are done training.
- Calculate the fitness of each neural network so that we can choose the parents for the next generations. The higher the fitness function higher the chance of getting selected.
- Find the maximum fitness in the population.
- Pick two neural networks based on a probability system regarding their fitness.
- Crossover the genes of the two neural networks. This will create a ”child” neural network. This neural network should have some properties of the first neural network and some of the second. The genes of the ”child” network are mutated. This is so that we can maintain randomness in the GA.
- The genes of the ”child” network are mutated. This is so that we can maintain randomness in the GA.
- Perform steps 5 — 7 for the number of neural networks in the population. Replace the old population with the newly generated population.

C. Weight Agnostic Neural Networks

In this paper [3], the proposed algorithm uses the weight agnostic properties and prefers simplicity. This algorithm is an evolutionary search in the space of possible architectures. The following is a gist of how the algorithm works.

- First, we need to create a set of minimal neural networks that can be embedded in single-parent architectures. No weights are assigned. The networks are just connections. In other words, each network is equivalent to a specific prune of the parent.
- Test the networks with a set of weights that are shared. The wights are shared so that the algorithm can drive a search that is weight agnostic.
- Rank the networks based on their performance and complexity. Since the networks shared weights, their performance reflects weight agnostic capabilities.
- Create new networks by taking the top-ranking networks and randomly modifying them.
- take the obtained networks and go to Step 2. The performance of the classifications methods was measured using recall, precision, F-measure. According to their experiment, the Na¨ıve Bayes classification proves to be the most efficient among the three algorithms for text classification of opinion mining.

III. TECHNOLOGIES

A. React

React [4] is a JavaScript library used to create interactive single-page user interfaces.

React is a JavaScript library used to create interactive single-page user interfaces.[4] The reason we chose react is that it makes the building of interfaces very easy. We can design simple views for each component, and when the data changes, the framework will render just the right components (according to the changes in the data). In react, it’s easier to manage complex UIs since each component can maintain its own state. Since react is a JavaScript library, it can be imported into any type of codebase.

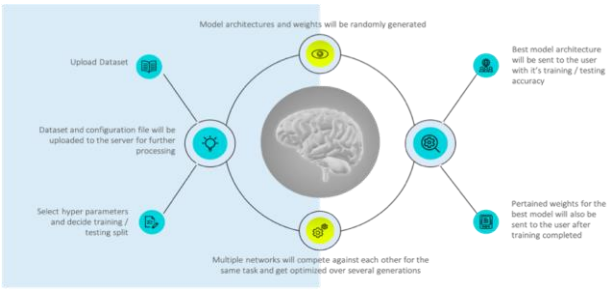

Fig. 1. Application Model

B. Flask

Flask [5] is a lightweight WSGI web application framework. The main reason why we chose Flask is that the black box of our code was built using python libraries. Flask allowed us to create the bridge between the web frame of our application to main the logic. Because the framework is light, it did not cause our servers to be burdened with anything that was not needed.

C. Tensorflow

TensorFlow [6] is an end-to-end open-source platform for machine learning. Tensorflow has APIs that support the creation of basic neural network architecture to complex deep convolutional neural networks. The Tensorflow APIs also harness the power of GPUs to make the processing of training of networks faster. Since our application asks for the training data from the user, we assume that the data we will get is going to be huge, and so we will need the power of the GPU to boost our code’s performance.

IV. SYSTEM MODEL

The frontend that is built using react takes the user input that consists of hyperparameters, the user’s email address, and the input files that consist of training and the testing dataset. Once the server receives the input from the users, we split the data into training and testing samples and, using our algorithm that is explained below, and we get the final backpropagation neural network.

We send an email to the user with the architecture of the neural network as a JSON and the whole set of weights that gave us the best accuracy. Using these details, the user can regenerate the neural network.

V. METHODOLOGY

For optimising and training our neural networks the steps will be as follows:

- Importing the uploaded dataset.
- Data preprocessing

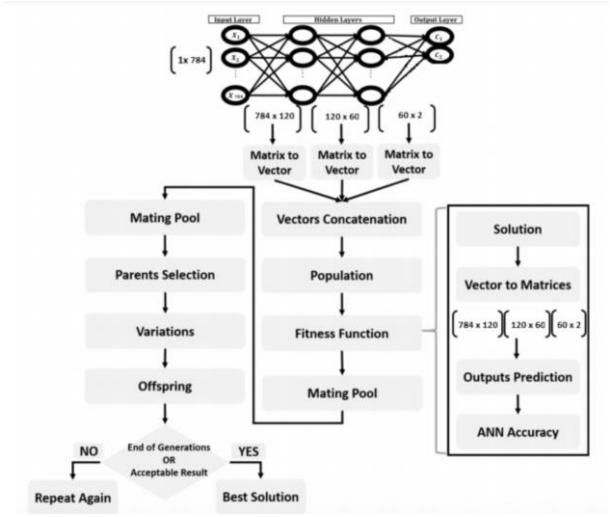

Fig. 2. System overflow diagram

- We then create a population of Artificial Neural Network using TensorFlow.
- We will train our ANN using Genetic Algorithm
- We then ranked our networks based on their performance and then we created offsprings. For the architecture we choose the architecture from the parent which has the minimum number of epochs and layers.
- We then repeat the process again for the specified number of generations Once all the generations have been trained we email the best network architecture and the trained weights to the user.

A. Artificial Neural Network

The human brain is basically a cluster of interconnected neurons that have inputs and outputs. Artificial neural networks are built to emulate the same process. We create layers of hidden nodes that generate as per the inputs and outputs provided to the networks. These neurons are often referred to as processing units. A neural network may have thousands of neurons. For our case (considering the mnist dataset), there will be 784 input units for each pixel of the image in the mnist dataset, and we will have ten output neurons. The ann can have between 2 - 10 hidden layers. Each layer can have between 2 - 10 hidden nodes and a random activation function. For our experiment, we will randomly initialize the weights for our ann and then train it using two different techniques Genetic Evolutionary Algorithm and Backpropagation.

B. Training Artificial Neural Network using Genetic Evolutionary Algorithm

Each step involved in the GA has some variations. The are several components to a genetic algorithm they are as follows:

- **Gene:** GA works on a population consisting of some solutions where the population size is the number of solutions. Each solution is called an individual. Each individual solution will have chromosomes. Four our algorithm, each chromosome is represented as a set of parameters (features) that defines the individual. Each chromosome has a set of genes. For our case, the genes will represent the weight of the networks, the number of layers in the network, the number of hidden nodes in the network, and the activation function used for the layer.

$$Accuracy = \frac{NumCorrectClassify}{TotalNumSamples}$$

Fig. 3. System overflow diagram

- **Fitness Function:** The fitness function will provide us with the fitness value that will let us know the quality of the solution. As a rule of thumb, we assume higher the fitness value will be, the higher will be the quality of the solution. For our case, the fitness function will simply be the accuracy achieved by the solution. This can be calculated by the formula given below:
  The fitness will be simply the validation accuracy returned by the TensorFlow model after training the algorithm using backpropagation. Which can be calculated by the formula given in figure 3.
- **Parent Selection / Mating Pool:** The fittest individuals will be selected for mating based on their fitness score. The individuals with the highest score have a higher probability of getting selected for mating. These individuals are called parents. We choose two parents to create two offspring, and The offspring will have 50% genes from each parent. As we select better-performing individuals, there will be a higher probability that the results produced will also be better and keep on improving. Finally, this will end up with the desired optimal or acceptable solution.
- **Generation:** All individuals are a part of a generation. Each training epoch, a new generation is created, which includes the best performing individuals from the previous generations and the new offspring created from them. The process of replacing the old population with the new one is called replacement. For our training, we will have the individuals train for five generations or until they achieve the best possible solution.
- **Crossover:** Crossover in GA generates new generations by mutating the old generation parents, the new generation offspring come by carrying genes from both parents. The amount of genes carried from each parent is random. In our case the offspring will take half of its genes from one parent and the other half from another parent.
- **Mutation:** Crossover in GA generates new generations by mutating the old generation parents. The new generation offspring come by carrying genes from both parents. The amount of genes carried from each parent is random. In our case, the offspring will take half of its genes from one parent and the other half from another parent.

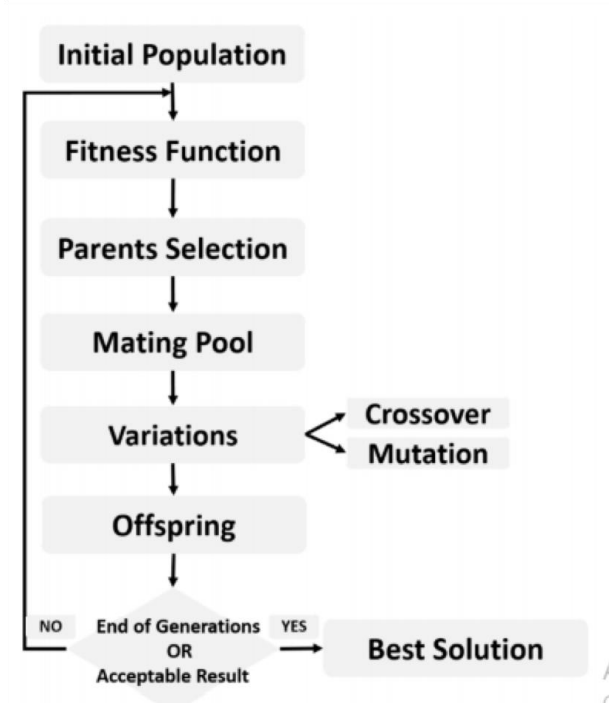

Fig. 4. System overflow diagram

C. Training Artificial Neural Network using Backpropagation

To train our model, we will use the TensorFlow library's sequential model. As implementation of the backpropagation algorithm is not our primary focus for this research, and it has been explored extensively in the past. We can simply reuse previous research on the subject. The primary objective of backpropagation is to optimize the loss function, e.g., gradient descent, and fine-tune the weights of the model accordingly over several iterations or epochs. Our model will be training for 100 epochs in both cases.

VI. CONCLUSION

As we can see, there can be many conclusions drawn from these results. The training time is a bit high, but the initial goal of making the training of neural networks easier for the user has been achieved. Also, we can see that the model can achieve minimalistic architecture with very high accuracy due to backpropagation and genetic algorithm working in combination. The User Interface makes it very easy for the user to upload and start training their networks. The emails sent to the user after completion of the training containing the network architecture and the trained weights save a lot of time for the user. But we can also see that due to large training times, larger datasets may take much longer to train. As a result, parallel training is required as each individual can be trained separately make the algorithm exponentially faster.

VII. FUTURE WORK

Several changes can be made in the future to the algorithm and user interface, such as:

Support for different networks

Support for multiple types of networks such as CNN, RNN, and LSTMs can be added by making modifications to the current layout.

Worldwide Access

Deploy the application on the internet using a hosting service and several virtual machines. This will enable the user to access the system from anywhere in the world.

Multiple Layer Support

Currently, only sequential layers are being optimized, but in future optimization for different types of layers such as dropout layers or batch normalization can also be added.

Informative UI

The UI currently has no progress indication or easy way to visualize the dataset uploaded this, and many other features can also be added.

REFERENCES

[1] Distilling the Knowledge in a Neural Network by Geoffrey Hinton, Oriol Vinyals, and Jeff Dean.Authors and Affiliations

[2] Castillo, Pedro Arenas, M.G. Castillo-valdivieso, J. Merelo Guervos, Juan Prieto, Alberto Romero, Gustavo. (2003). Artificial ´ Neural Networks Design using Evolutionary Algorithms

[3] The lottery ticket hypothesis: Finding Sparse, Trainable Neural Networks by Jonathan Frankle, Michael Carbin

[4] "React - A JavaScript library for building user interfaces". React. Retrieved 7 April 2018.

[5] Grinberg M. Flask web development: developing web applications with python. " Ox27;Reilly Media, Inc." 2018.

[6] Mart ´ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org