An Atypical Metaheuristic Approach to Recognize an Optimal Architecture of a Neural Network

Abishai Ebenezer M. and Arti Arya[a]
PES University, Bangalore, India

Keywords: Artificial Neural Networks (ANNs), Structural Design, Water Cycle Algorithm (WCA), Water Wave Optimization (WWO), Metaheuristic Algorithm.

Abstract: The structural design of an Artificial Neural Network (ANN) greatly determines its classification and regression capabilities. Structural design involves both the count of hidden layers and the count of neurons required in each of these hidden layers. Although various optimization algorithms have proven to be good at finding the best topology for a given number of hidden layers for an ANN, there has been little work done in finding both the optimal count of hidden layers and the ideal count of neurons needed in each layer. The novelty of the proposed approach is that a bio-inspired metaheuristic namely, the Water Cycle Algorithm (WCA) is used to effectively search space of local spaces, by using the backpropagation algorithm as the underlying algorithm for parameter optimization, in order to find the optimal architecture of an ANN for a given dataset. Computational experiments have shown that such an implementation not only provides an optimized topology but also shows great accuracy as compared to other advanced algorithms used for the same purpose.

1 INTRODUCTION

As humans, it is quite natural to be inspired by nature and carry that inspiration , irrespective of the domain, into building our own designs and structures. Researchers in computer science, and specifically, deep learning, have been inspired by nature as well and have long been trying to build models that exhibit or replicate what the human brain can accomplish. Our brain contains millions of neurons, and what is more interesting is that these neurons are assembled perfectly in order for us to make both simple and complex decisions. Deep learning practitioners are often confronted with the dilemma of having to manually design a structure while building an ANN. With the advent of various kinds of bio inspired algorithms, it is now possible to draw inspiration from nature while building an ANN wherein the main goal is to find the global optimum. This could also be applied to the problem of structural optimization where the search space consists of all the possible ANN structures. On earth, water is known to flow downwards in its liquid form due to gravity. Therefore, streams flow into rivers while rivers flow into the sea. The sea is synonymous to global minimum/optimum. Water from the sea then evaporates forming clouds followed by the raining process which leads to formation of streams and the cycle goes on. For minimization problems, the water cycle provides a natural analogy. And just as all streams and rivers flow into the sea, agents employed in finding a global minimum in the search space could be made to follow a similar movement.

The human brain has approximately 86 billion neurons (Azevedo et al., 2009) assembled perfectly to perform complex computations and make fast decisions, demonstrating that the structure of an ANN goes a long way in determining its accuracy and efficiency. Those who build and create ANNs are often at a loss to know what the structure of their ANN must be. After numerous trial and error manual iterations, the final structure that they adopt would most probably, not be the most optimal architecture. If more hidden layers or more neurons are used than needed, it would lead to excessive training time and faulty results due to overfitting, especially when the training data set is large. When the number of neurons and hidden layers used is lesser than required, it would lead to underfitting and the features present in the data would not be learnt by the ANN. An optimal structure means efficient use of compute resources and precise learning of features present in the data for ANNs.

[a] https://orcid.org/0000-0002-4470-0311

This paper proposes a novel approach to cope with the variable dimensions of the structure optimization using a combination of the Water Cycle Algorithm (WCA) for structure optimization and back-propagation for parameter training (Eskandar et al., 2012). The WCA has been applied to two levels of search spaces - the global search space and the local search space of each particle in the global population, in order to return the most optimal structure for an ANN given a particular dataset. The primary goal of this proposed methodology is to do better and improve upon the existing approaches, while also proving the fact that bio-inspired metaheuristics can effectively be used in the field of Neural Architecture Search.

2 LITERATURE REVIEW

Methods used for Neural Architecture Search could broadly be classed into two - traditional methods and non-traditional methods. Traditional optimization methods have predominantly been calculus based. Guliyev et al.(Guliyev and Ismailov, 2018b) proved that a neural network having two neurons in one hidden layer, and all weights having a fixed value equal to 1, could accurately approximate any continuous univariate function. The disadvantage of the proposed methodology is that it could not approximate all continuous multivariate functions. The same authors then overcame this drawback by proving that, by considering the Euclidean Space to consist of d dimensions and by using a neural network having two hidden layers, wherein the count of hidden neurons it contains is $3d$+2, any continuous d-variable function could be correctly approximated(Guliyev and Ismailov, 2018a).

Ludwig(Molino et al., 2019) is an AutoML model created by Uber Research, used for deep learning model training by passing their data through an API. Ludwig uses an encoder-combiner-decoder model and acts like a black-box which abstracts the process of model training. Although Ludwig achieves great results, researchers still look for solutions to find the optimal structure of a deep feed-forward neural network which would enhance our understanding and perception of the underlying working of deep feed-forward neural networks.

Cai et al. suggested a technique to ascertain the ideal count of neurons required in a neural network having just one hidden layer, by using data normalization and singular value decomposition(Cai et al., 2019). The count of hidden neurons would then be determined by the main eigen values. Pablo et al. suggested a non-iterative method for finding the optimal count of neurons in a singly hidden layered neural network while pruning the hidden neurons with respect to harmonic mean(Henríquez and Ruz, 2018). Hence, it is observed that although traditional methods have been able to achieve an acceptable level of performance for finding the optimal count of neurons in shallow neural networks, it still fails to resolve the issue of ascertaining both the ideal/near optimal count of hidden layers as well as the ideal count of hidden neurons for a deep neural network.

Neural Architecture Search has been an evolving field of research, especially in the recent past. A survey on various approaches to this problem was done by a team of researchers from IBM Research and is presented in (Wistuba et al., 2019). This survey presented a comprehensive study of the different kinds of search spaces that have been adopted and many traditional and non-traditional approaches have been explored. The survey also presented the various categories of optimization methods that have been used to traverse and explore the search space. Although the survey does not touch upon the use of bio-inspired algorithms, the authors have elaborately specified contributions belonging to the class of evolutionary algorithms in the field of Neural Architecture Search.

Bio-inspired algorithms simulate biological evolution in nature and have proven to show remarkable results for solving complex optimization tasks. Kennedy and Eberhart created the first ever bio-inspired algorithm - Particle Swarm Optimization (PSO)(Kennedy and Eberhart, 1995). PSO takes inspiration from the way a flock of birds search for food. PSO has been a popular choice for bio-inspired algorithms, especially due to its easy-to-understand methodology and simple implementation. Similarly, researchers have developed bio-inspired algorithms that simulate the collaborative characteristics of schools of fish(Zhang et al., 2014), ant colonies(Dorigo and Di Caro, 1999) and even bats(Gandomi et al., 2013). These algorithms consider properties specific to the biological species and translates them into equations. These new equations then give rise to a bio-inspired algorithm which can then be used in various engineering optimization tasks. PSO has also been used for both parameter optimization(Qolomany et al., 2017) and structure optimization(Chhachhiya et al., 2017). Due to its highly simplistic nature, PSO falls short in terms of the final results obtained.

The Water Wave Optimization (WWO) is a relatively recent bio-inspired optimization algorithm, developed in 2015(Zheng, 2015). This algorithm takes inspiration from shallow water wave models, wherein the solution is represented by a wave. It involves the use of three different operators - namely propagation, refraction, and breaking. Zhou et al. proposed a methodology to use the WWO algorithm for simultaneously solving both parameter optimization and structure optimization(Zhou et al., 2018). Although the algorithm did achieve competitive performance, it did have a few drawbacks. For parameter optimization, the WWO algorithm was much slower compared to the back propagation algorithm. One of the main disadvantages in the application of WWO to the problem of structure optimization is that the count of hidden layers has to be preset by the user before the start of computation. The problem of structure optimization, where the optimal count of hidden layers and the optimal count of neurons required in each of the hidden layers, still remains to be solved in a manner wherein the user themselves does not have to specify the exact count of hidden layers.

The WCA, developed in 2012, inspired by the water cycle on earth(Eskandar et al., 2012), presents itself as a natural choice for an optimization algorithm that can be used in multidimensional space. Water at higher level flows to a lower level, under the influence of gravity and this analogy is especially suited for the minimization paradigm for the specific engineering problems. The combination of WCA, backpropagation and the use of both global and local search spaces, has proven to show competitive results in various constrained engineering optimization problems and upon being used in the proposed approach in this paper, shows great promise in solving the structure optimization problem for deep neural networks.

Poongodi et al. (Poongodi et al., 2021) presented a trained XGBoost model that was able to predict the taxi trip durations having an RMSE value of 0.39, and concluded that XGBoost performed better than a Multi-layer Perceptron (MLP) model. Although the exact MLP structure that they have used is not exactly known, the structure evidently was not an optimal structure, as shown in section 5.

3 THE WATER CYCLE ALGORITHM

As suggested by its name, the process of water cycle is the main motivation behind the WCA(Eskandar et al., 2012). This algorithm captures the flow of rivers and streams downstream into the sea. The sea represents the global minimum. This is because streams flow into rivers, which then flow into the sea which is the universal lowest region. To begin with, the raining process occurs, which leads to the formation of streams. The raining process creates the initial population of streams in a random manner. The best stream of the initial population is selected as the sea. In case of a minimization problem like the problem of structure optimization for ANNs, this means to search for the stream with the highest fitness value i.e. the stream with the least value of cost function (mean squared error).

Subsequently, a number of good streams (other than the sea) are chosen as rivers. This number is one of the parameters given by the user, and its value can be decided on the basis of the extent of exploration required. The other streams flow towards the rivers or the sea.

The term particle in context of neural architecture, refers to an individual object that collaborates with other similar objects to achieve a common objective. In the case of the WCA, particle and stream are equivalent and may be used interchangeably.

The initial step of creating the population of streams i.e. the raining process, involves creating an array of randomly generated integers in a given range. These integers represent the particle's location defined in the search space. The range of the randomly generated integers by the algorithm is another parameter required from the user. Let's assume the population size is N_{pop}. Hence, an array with N_{pop} elements would be created, each specifying a randomly generated integer in the given range representing position of the stream/particle in the search space. The fitness value of each stream can be found using a fitness function F. Considering a specific element i.e. an integer representing a particular stream to be X, $F(X)_n$ would then return the fitness value of the particular stream X, where n simply signifies that X is the nth stream in the population.

The sea represents the best stream i.e. the best position reached so far in the search space. Therefore, the sea is selected as the best individual i.e. the sea is the element which has the least value of cost function in the array, which means that at all times, the element having the highest fitness value would be chosen as the sea. Consider a variable N_{SR}, which is the count of rivers (as provided by the user) in addition to the single sea. Also, $N_{streams}$ is defined as the size of the remaining population i.e. the streams that are not rivers or the sea.

The quantity of water that flows into a particular river or sea varies for each river and the sea. The flow of water into each river or the sea is defined by a variable NS_n and can be calculated using the below equations as described in (Eskandar et al., 2012):

$$N_{\text{streams}} = N_{\text{pop}} - N_{\text{SR}} \quad (1)$$

$$NS_{\mathrm{n}} = round(\left|\frac{F(X)_{\mathrm{n}}}{\sum_{i=1}^{N_{SR}} F(X)_i}\right| \times N_{streams}) \quad (2)$$

n = 1,2,....,N_{SR}

$$p_{Str}^{i+1} = p_{Str}^{i} + rand * C * (p_{R}^{i} - p_{Str}^{i}) \quad (3)$$

$$p_{Str}^{i+1} = p_{Str}^{i} + rand * C * (p_{Sea}^{i} - p_{Str}^{i}) \quad (4)$$

As mentioned in (Sadollah et al., 2016), Equation (3) represents streams that would flow into the corresponding rivers and Equation (4) represents streams that would flow into the corresponding sea.

$$p_{R}^{i+1} = p_{R}^{i} + rand * C * (p_{Sea}^{i} - p_{R}^{i}) \quad (5)$$

p^{i}_{Str} represents the position of the particular stream at the *i*th timestep. *Str* signifies a particular 'Stream' and *R* signifies a particular 'River'. $\mathrm{NS_n}$ is the count of streams that would flow into that particular river or sea.

In order to avoid premature convergence i.e. to avoid getting stuck in a local minimum, an evaporation operation is defined which involves the evaporation of sea water. The evaporation process is enabled when the river and the sea are separated by a very small distance *d* i.e. when the below criteria is satisfied.

$$\left|p_{Sea}^{i} - p_{R}^{i}\right| < d \quad (6)$$

i = 1,2,3,...,N_{SR} - 1

d is a near zero number, and d is an important factor in the exploration phase of the WCA. If the evaporation operation as in Equation (6), is satisfied, the raining process is then performed, which thus gives rise to new streams in the population, which are initialised randomly. The value of d is determined at every time step using the following equation -

$$d^{t+1} = d^{t} - \frac{d^{t}}{m} \quad (7)$$

t = 1,2,3,...,*m*. *m* is the maximum count of iterations.

A higher value of *d* signifies lower search intensity while a lower value of d encourages the intensity of exploration closer to the sea.

This entire water cycle procedure is then repeated for as many iterations as required in order to identify the most optimal region/point in the search space.

The motivation towards adopting the WCA for finding the optimal architecture of an ANN is its simplicity and ease of implementation. It was proposed as a general solution for constrained optimization and engineering design problems. Its functioning, which is easily understandable as it is derived from the water cycle, is highly relatable to find the global minimum in multidimensional space. The novelty of the proposed methodology elevates the level of performance, as shown in the subsequent sections, thus stressing more on the proposed approach.

4 PROPOSED APPROACH TO ARCHITECTURAL OPTIMIZATION FOR ANNs

The approach suggested in this paper involving the use of global and local search spaces, the WCA algorithm, as well as the back-propagation algorithm is a significant upgrade compared to approaches taken in the past. Not only does the unconventional approach proposed in this paper solve the problem of optimizing the count of neurons in each of the *n* hidden layers but also uses the WCA algorithm to find out the ideal value of *n* itself.

The global search space represents the error as a function of the count of hidden layers. The fitness of each stream in the main space is determined by the least value of cost function(error) that can be obtained by using that particular number of hidden layers. Each particle contains the information as to the least possible error and the corresponding configuration i.e. the count of hidden layers and the count of neurons required in each of the hidden layers. The central focus in this section is the approach that orchestrates the movement of these particles in both the global search space as well as every local search space.

The WCA is applied at the global and local levels for the structure optimization problem (as shown in Figure 1). As mentioned earlier, two types of spaces would be considered to cope with variable dimension problem. The global search space represents the count of hidden layers. The population of streams/particles would then search this space to determine what would be the best number of hidden layers. Considering that the initial population of streams would be used to ascertain the ideal count of hidden layers required, each stream in itself represents a search space (local) which contains all the possible configurations for that particular count of hidden layers as represented by the stream. WCA is again applied in this local search space of each stream to find out the best configuration i.e. the optimal count of neurons in each of the hidden layers for the specific number of hidden layers (this number is the parent stream in the global space).

Having explained why the WCA has been chosen as a part of this study, the reason why back-propagation seems a more suitable choice in the lo-

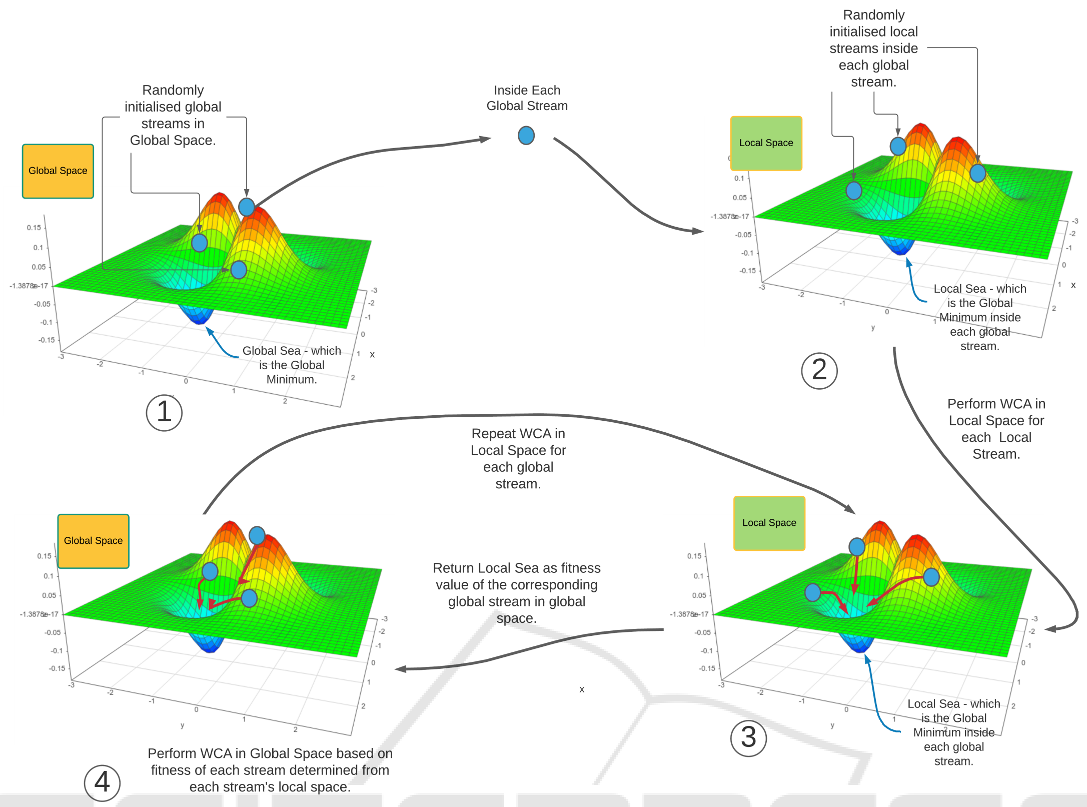
Randomly initialised global streams in Global Space.
Global Space
Global Sea - which is the Global Minimum.
1
Inside Each Global Stream
Randomly initialised local streams inside each global stream.
Local Space
Local Sea - which is the Global Minimum inside each global stream.
2
Perform WCA in Local Space for each Local Stream.
Repeat WCA in Local Space for each global stream.
Global Space
Return Local Sea as fitness value of the corresponding global stream in global space.
Local Space
Local Sea - which is the Global Minimum inside each global stream.
3
Perform WCA in Global Space based on fitness of each stream determined from each stream's local space.
4

Figure 1: The proposed approach using WCA for optimal architecture search.

cal search space to train an ANN for parameter optimization must be mentioned. The entire search space, which includes the global search space and all the local search spaces put together, is very large, and hence, the optimization algorithm used must be able to explore the local search spaces in a swift manner. Although the WCA is sufficiently efficient, it isn't the fastest optimization algorithm. Therefore, to enhance the performance of the system, backpropagation has been used for parameter optimization. Backpropagation is one of the most preferred algorithms for parameter optimization due to its easy implementation and comparatively lesser run-times.

The proposed novel approach involves applying the WCA algorithm at two levels (as shown in Figure 2) - the global search space and local search spaces, while using back-propagation to train the ANN i.e. for parameter optimization. A population of streams is created randomly, wherein each stream contains a randomly generated integer value (within a range specified by the user) which represent the count of hidden layers. The goal now is to assign a fitness value to each stream. The fitness value of each stream would represent how close or far the particular stream is from its final goal. The fitness value of each stream in the global search space is determined by the value of the cost function of the best configuration for that particular number of hidden layers, as represented by the stream in the global search space. For example, if a stream S contains the integer X, the fitness value assigned to this stream would be the value of cost function of the best configuration containing the optimal count of neurons in each of the X hidden layers. Therefore, for a stream to be considered a 'good' stream (like a river or the sea), then it must have a higher fitness value i.e. a lower value of cost function. The sea would have the highest fitness value followed by the rivers and so on.

The goal now is to find the 'best configuration' for X hidden layers. The local search space of each stream can be intuitively visualized as a search space within the stream in the global search space i.e. a search space within a search space. The local search space represents the entire set of possible configurations for X layers. WCA is applied in this local search space as well. An initial population of streams is created, wherein the streams represent an array of integers. The ith element in the array represents the count of neurons in the ith hidden layer. For example, consider an array A = (A_1, A_2,...,A_n), where A_i is the number of neurons in the ith hidden layer. Unlike the global search space which initially seemed only to be

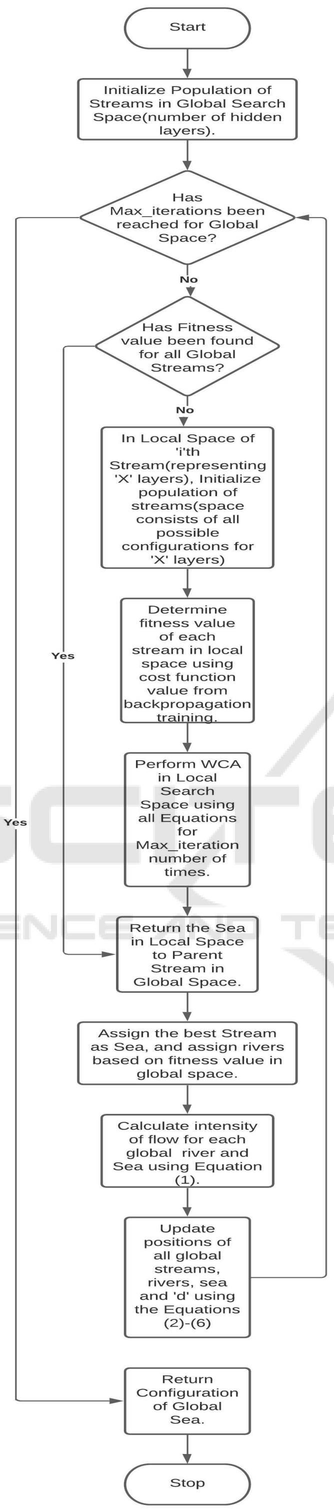
Start
Initialize Population of Streams in Global Search Space(number of hidden layers).
Has Max_iterations been reached for Global Space?
No
Has Fitness value been found for all Global Streams?
No
In Local Space of 'i'th Stream(representing 'X' layers), Initialize population of streams(space consists of all possible configurations for 'X' layers)
Determine fitness value of each stream in local space using cost function value from backpropagation training.
Perform WCA in Local Search Space using all Equations for Max_iteration number of times.
Yes
Yes
Return the Sea in Local Space to Parent Stream in Global Space.
Assign the best Stream as Sea, and assign rivers based on fitness value in global space.
Calculate intensity of flow for each global river and Sea using Equation (1).
Update positions of all global streams, rivers, sea and 'd' using the Equations (2)-(6)
Return Configuration of Global Sea.
Stop

Figure 2: Flowchart of Proposed Approach.

a single dimension problem, the local search space is an X dimensional problem (X is used here in the same sense as mentioned previously), where X is an integer in the range specified initially by the user.

Algorithm 1: Applying WCA for Architecural Optimization.

```
Initialize a population of streams $Global_{Streams}$ randomly;
while Max.Iterations for Global Space has not been reached do
    for all $S \varepsilon Global_{Streams}$ do
        Initialize a population of streams $Local_{Streams}$ randomly;
        while Max. Iterations for Local Space has not been reached do
            Determine fitness of each stream in $Local_{Streams}$;
            Update the position of each stream using Equations (1)-(6);
            Update position of Local Sea of S;
        end while
        return Stream with maximum fitness in $Local_{Streams}$ as fitness of S;
    end for
    Update position of all streams in $Global_{Streams}$ using the equations;
    Update position of Global Sea;
end while
return Global Sea and its Local sea
```

During the step of calculating the new values of position of both streams and rivers using Equations (1) to (6), the stream is considered to be an X dimensional particle. The cost function i.e. the fitness value of all streams would be determined by the cost function of an optimally trained ANN (using back-propagation with a necessary and sufficient amount of training epochs) with hidden layers as specified by the array contained in the particular stream. The WCA process is then repeated for a particular number of iterations, as specified by the user. The stream with the highest fitness value i.e. the sea present in the global search space is returned. Therefore, the stream in the global search space contains 3 different values - the number of hidden layers, an integer array of size X (obtained from local search space) and the fitness value which is the cost function of the ANN sufficiently trained using back-propagation (obtained from the local search space). In the global space, only the count of hidden layers is used to update the positions of the rivers and streams. This process is repeated until the desired accuracy is achieved. Upon conclusion of the final iteration in the global space, the stream

Table 1: Results on New York Taxi Trip Duration Dataset.

Neurons in each hidden layer	Number of hidden layers	RMSE	Average	Min	Max
[300, 113, 1, 88, 101]	5	0.3751	0.3833	0.3751	0.396
[139, 223, 42]	3	0.3848			
[218, 126, 157, 75]	4	0.3799			
[255, 157, 173]	3	0.3786			
[183, 61, 99, 265, 218, 296, 139, 265, 286, 51]	10	0.3911			
[224, 94, 128, 272, 47, 177, 137, 1]	8	0.387			
[218, 242, 81, 131, 155, 242, 82, 275]	8	0.3793			
[57, 84, 42, 142, 195, 50, 124, 140, 191]	9	0.396			
[245, 300, 230, 108, 110, 138, 1, 37, 24, 276]	10	0.3764			
[106, 270, 87]	3	0.3845			

Table 2: Results on the WINE Dataset for WCA as compared to WWO.

T:V	Metric	GA	PSO	BBO	WWO	WCA(1)	WCA(2)
2:3	Max	84.11	94.39	86.92	98.13	95.28	98.11
	Min	79.44	93.46	83.18	96.26	94.34	90.57
	Mean	81.7	93.69	84.89	96.65	94.528	94.85
3:2	Max	84.87	94.12	88.24	97.48	98.59	98.59
	Min	82.35	90.76	85.71	97.48	92.96	92.96
	Mean	83.6	92.23	86.86	97.48	95.212	96.16
4:1	Max	80.56	97.22	88.89	100	100	100
	Min	77.78	97.22	77.78	100	94.28	94.29
	Mean	79.32	97.22	81.38	100	98.284	98.44

with the highest fitness value i.e. the sea, which contains the optimal count of hidden layers, ideal count of neurons required in each hidden layer and its fitness value is returned to the user. Algorithm 1 describes the application of WCA for architectural optimization for ANNs.

5 RESULTS AND DISCUSSION

The WCA is a simple yet intuitive optimization metaheuristic algorithm(Eskandar et al., 2012). Other bio-inspired algorithms have also been developed, amongst which the WWO algorithm has been able to achieve state-of-the-art results(Zhou et al., 2018).

The proposed model was executed on the New York Trip Duration dataset [2] 10 times, and the results obtained have been shown in Table 1. The configuration of the proposed model was the same as that used for the previous experiments. The New York Trip Duration dataset consists of taxi trips data such as pickup time and date, drop off time and date, geo-coordinates of pickup and drop off, number of passengers and several other variables, wherein the duration of each taxi trip is to be predicted. A sample of one million rows were taken out of the the 1458644 rows available in the training dataset. The range of number of hidden layers (input parameter) was configured to be between 1 and 10 and the count of neurons in each layer ranges between 1 and 300. Also, d, mentioned in Equation (6), is set to 0.01 for all the performed experiments, as suggested in (Sadollah et al., 2016). Table 1 shows that the proposed methodology is able to find various optimal architectures for this dataset. 8 out of the 10 proposed structures have an RMSE value below 0.39, which is slightly better than the XGBoost model presented in (Poongodi et al., 2021) - previously considered to work better than an MLP model. This shows that the proposed approach is able to find optimal architectures for neural networks whose performance can match and even better those of other supervised machine learning algorithms, even when other ma-

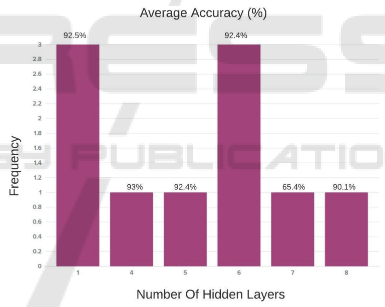

Figure 3: Frequency and Average Accuracies on all the suggested count of hidden layers for the Phishing Websites Dataset(Vrbančič, 2020).

[2]https://www.kaggle.com/c/nyc-taxi-trip-duration/data

Table 3: Results on Phishing Websites Dataset for 1 and 6 Hidden Layers.

Neurons in Each Hidden Layer	Number Of Hidden Layers	Accuracy (%)	Average	Max
[87,48,55,59,67,100]	6	93.12	92.43	93.12
[30,28,52,98,49,65]	6	92.03		
[100,78,74,40,20,57]	6	92.14		
[78]	1	94.03	92.50	94.03
[76]	1	91.82		
[100]	1	91.66		

chine learning learning algorithms were previously perceived to perform better than an MLP model, as is the case here.

The WINE Dataset (Dua and Graff, 2017) is a popular dataset based on classification output. The main task here is to correctly classify the category of wine based on parameters such as color, acidity etc. The proposed algorithmic approach was executed 30 times (30 was decided randomly, for the main purpose of presenting how well the proposed method performs over multiple executions), while the range of number of hidden layers (input parameter) was between 1 to 10 and the count of neurons in each layer ranges between 1 and 100 for the remaining performed experiments. Configurations having 2 hidden layers was suggested 11 out of the 30 times it was executed, proving that the proposed approach works well for such benchmark datasets.

The proposed approach performs at par with the WWO algorithm, and has performed better than algorithms like Particle Swarm Optimization (PSO), Genetic Algorithms (GA) and Biogeography-based optimization (BBO) and finds the optimal count of hidden layers as well as optimal count of neurons in each hidden layer, and hence, specifying the number of hidden layers beforehand is not a requirement. In Table 2, the first column T:V represents the ratio of size of training dataset to validation dataset. The results shown have been obtained using the WINE dataset. As stated in (Zhou et al., 2018), the time taken by backpropagation for parameter optimization in the local search space is least, and hence serves as the motivation for using backpropagation in the proposed approach. WCA(1) and WCA(2) represent two different configurations. WCA(1) was configured with 6 streams in the global space and 4 streams in each local space. WCA(2) was configured with 5 streams in the global space and 5 streams in each local space. Both were configured to suggest the number of hidden layers within the range of 1 to 10 and the count of neurons in each layer between 1 and 100.

As seen from Table 2, the proposed approach using WCA falls slightly short as compared to the WWO. This is not to be mistaken that the proposed approach is in any way inferior to the WWO. The main shortcoming of the approach that uses the WWO, is that the number of hidden layers has to be specified by the user beforehand, whereas this is already taken care of in the proposed approach using WCA. The fact that the proposed approach suggests both the count of hidden layers as well as the count of neurons in each hidden layer, while maintaining an accuracy that is almost at par with a state-of-the-art bio-inspired approach like WWO and achieving an accuracy better than older approaches such as GA, PSO and BBO shows the superiority of the proposed methodology.

It was also tested on the the Phishing Websites Dataset(Vrbančič, 2020)[3]. Generally, while creating an ANN for a dataset having thousands of rows and a large number of features to be learnt, ANN creators naturally use a large count of layers and neurons. Such an extensive architecture may not always be necessary, as proved by the following experiment. The Phishing Websites Dataset contains 88647 rows and 112 columns. As depicted in Figure 3, the proposed approach was executed 10 times for this dataset and both one hidden layer and six hidden layers were suggested 3 times each, whereas the remaining number of hidden layers were suggested only once. Although the number of features to be learnt are high, on executing the approach proposed in this paper, it was found that one hidden layer, containing 78 neurons was able to achieve an accuracy of just over 94%, using 3000 training epochs and at a learning rate of 0.001. Table 3 shows the results obtained for one and 6 hidden layers with the mentioned number of neurons in each hidden layer. This goes on to show that this proposed approach not only works well with small datasets, but can easily be scaled to large real world datasets as well, which is easily achievable given the power and capabilities of modern day computing hardware.

6 CONCLUSION

The novel approach proposed in this paper shows its capability in coping with the variable dimension problem of architecture optimization by achieving state-

[3]https://data.mendeley.com/datasets/72ptz43s9v/1

of-the-art results. This goes on to prove not only that metaheuristic algorithms are capable of being used effectively and accurately in multidimensional space, but also the use of global and local search spaces in finding the ideal/near optimal count of hidden layers (global) as well as the ideal/near optimal count of neurons in each hidden layer (local). Computationally performed tests confirm this by showing that the approach proposed in this paper achieves competitive results to other algorithms that have been developed lately, like the WWO algorithm, while also going one step further by finding the optimal count of hidden layers without the need of user intervention. As a part of extending the scope of this approach, progress is being made on adapting this methodology to more state-of-the-art algorithms and evaluating their performance. Also, efforts are being made to minimize the expanse of the overall search space.

REFERENCES

Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M., Ferretti, R. E., Leite, R. E., Filho, W. J., Lent, R., and Herculano-Houzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541.

Cai, G.-W., Fang, Z., and Chen, Y.-F. (2019). Estimating the number of hidden nodes of the single-hidden-layer feedforward neural networks. In *2019 15th International Conference on Computational Intelligence and Security (CIS)*, pages 172–176.

Chhachhiya, D., Sharma, A., and Gupta, M. (2017). Designing optimal architecture of neural network with particle swarm optimization techniques specifically for educational dataset. In *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, pages 52–57.

Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477 Vol. 2.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Eskandar, H., Sadollah, A., Bahreininejad, A., and Hamdi, M. (2012). Water cycle algorithm – a novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures*, 110-111:151–166.

Gandomi, A. H., Yang, X.-S., Alavi, A. H., and Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, 22(6):1239–1255.

Guliyev, N. J. and Ismailov, V. E. (2018a). Approximation capability of two hidden layer feedforward neural networks with fixed weights. *Neurocomputing*, 316:262–269.

Guliyev, N. J. and Ismailov, V. E. (2018b). On the approximation by single hidden layer feedforward neural networks with fixed weights. *Neural Networks*, 98:296–304.

Henríquez, P. A. and Ruz, G. A. (2018). A non-iterative method for pruning hidden neurons in neural networks with random weights. *Applied Soft Computing*, 70:1109–1121.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4.

Molino, P., Dudin, Y., and Miryala, S. S. (2019). Ludwig: a type-based declarative deep learning toolbox.

Poongodi, M., Malviya, M., Kumar, C., et al. (2021). *New York City taxi trip duration prediction using MLP and XGBoost*. Int J Syst Assur Eng Manag.

Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A., and Benhaddou, D. (2017). Parameters optimization of deep learning models using particle swarm optimization. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1285–1290.

Sadollah, A., Eskandar, H., Lee, H. M., Yoo, D. G., and Kim, J. H. (2016). Water cycle algorithm: A detailed standard code. *SoftwareX*, 5:37–43.

Vrbančič, G. (2020). Phishing websites dataset.

Wistuba, M., Rawat, A., and Pedapati, T. (2019). A survey on neural architecture search.

Zhang, C., Zhang, F.-m., Li, F., and Wu, H.-s. (2014). Improved artificial fish swarm algorithm. In *2014 9th IEEE Conference on Industrial Electronics and Applications*, pages 748–753.

Zheng, Y.-J. (2015). Water wave optimization: A new nature-inspired metaheuristic. *Computers & Operations Research*, 55:1–11.

Zhou, X.-H., Xu, Z.-G., Zhang, M.-X., and Zheng, Y.-J. (2018). Water wave optimization for artificial neural network parameter and structure optimization. In Qiao, J., Zhao, X., Pan, L., Zuo, X., Zhang, X., Zhang, Q., and Huang, S., editors, *Bio-inspired Computing: Theories and Applications*, pages 343–354, Singapore. Springer Singapore.