SHIV NADAR UNIVERSITY

Graded Assignment 1

Course Code: **CCC420** Course Title: **Nature inspired Computing**

Due Date: Feb, 17, 2020 **Max. Marks: 20**

Instructions

1. Solve the given problem in the space provided below question.
2. Upload the assignment on the black board.

Naming Convention for submission

Save this file using the following file naming convention: Graded_Assignment1_CCC420_Your-First-Name_Your-Last-Name.pdf. Be sure to use the underscores. For example, if your name is Doug Lea, this assignment file will be named as: **Graded_Assignment1_CCC420_Doug_Lea.pdf.**

Submitting this Assignment

You will submit (upload) this document to the same assignment area in Blackboard where you found it.

Grading Criteria

Write answers in your own words. Numerically correct and to-the-point answers will be awarded full points. **This assignment has 20 points. In the total of 100 points scale, this assignment will have 20 points.**

Q. Write a survey of all the Nature Inspired Algorithm discussed in this course in following parameters:

Brief Overview
Applications
Disadvantages
Advantages
Important Parameters that affect the algorithm
Extended Version of the algorithm in Literature
Future Work possible (Hybrid, Application areas, drawbacks overcoming solutions, etc.)

Contents

Introduction

Nature-inspired Algorithms (NAs) simulate the nature's way of solving its own problems [1]. Among the various classifications for nature-inspired algorithms are Stochastic or Deterministic, Bio-inspired, Environment-based or Physics/Chemistry-based, population-based or trajectory-based/single-solution based [1, 2] (figure 1).

Deterministic algorithms are designed to obtain the optimal solution but their **stochastic** counterparts cannot guarantee the optimal results but could solve larger and more complex problems within a relatively short time [1].

Bio-inspired algorithms (BAs) are inspired by the biological processes of nature such as the behaviour of living organisms, and their evolutionary trends. Bas may be classified as Evolutionary Algorithms, Swarm-based algorithms, Ecology-based algorithms. **Evolutionary algorithms** are broadly inspired by the Darwin theory of Evolution. Genetic Algorithms, Differential Evolution algorithms, and Genetic Programming are examples of evolutionary algorithms. **Swarm-based algorithms** are inspired by the cooperative or competitive behaviour of animals in our environment. Particle Swarm Optimization and Ant Colony Optimization are examples of the Swarm-based algorithms. **Ecology-based algorithms** are inspired by the natural ecosystem[1].

Nature-inspired stochastic algorithms could be to categorize them as either **population-based** or **trajectory-based/single-solution based**. Population-based optimization techniques solve problems using a number of search agents which also result in obtaining a number of solutions in a particular iteration. They also usually allow a selection of the best solution amongst the several generated solutions. Examples include genetic algorithms (GA), Particle Swarm Optimization (PSO). On the other hand, Simulated Annealing, Hill Climbing, Tabu Search are trajectory-based and use a single agent that moves through the search space in a zigzag fashion as the iterations continue. These algorithms usually obtain the initial solution randomly and then improved upon iteratively. Since these algorithms use a single search agent, they generate a single solution per iteration [1].

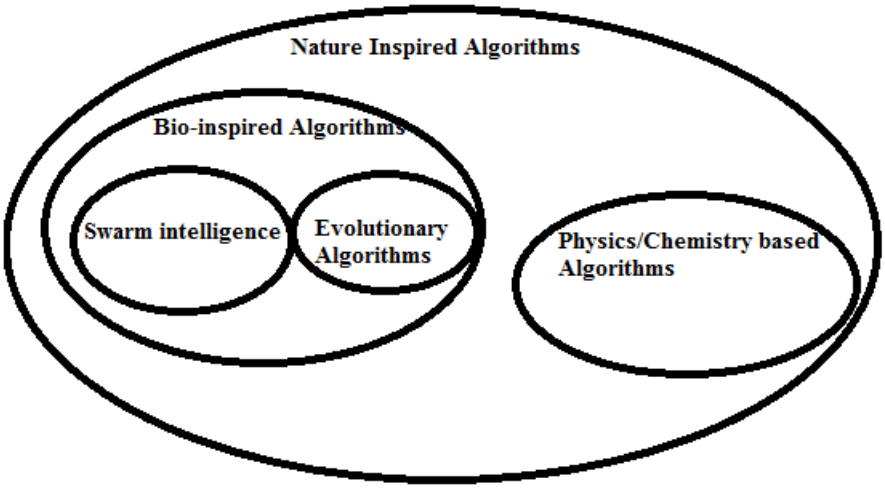


Figure 1 Venn diagram representing the various categories of NIAs [3]

Algorithms

```
1. Generate initial solution (s(0))
2. t := 0
3. While not Termination (s(t)), do
4. Explore neighborhood s'(t) := SelectMove(s(t));
5. If Move(s'(t)) accepted, then
6. s(t) := ApplyMove(s'(t))
7. End if
8. t := t+1
9. End While
```

Trajectory-based algorithm pseudocode [1]

```
1. Initialize population
2. Evaluate the objective function
3. Repeat
4. Evaluate the population quality
5. Apply the variation operator
6. Evaluate the objective function
7. Until termination condition
```

Population-based algorithm pseudocode [1]

Hill Climbing

Hill Climbing (HC), developed by Selman and Gomes [23], starts by selecting an arbitrary solution and then tries to obtain better solutions in an iterative manner by first by altering a single solution element. If the new solution is better, an alteration is made to another element of the new solution. The process is repeated until no improvement is possible on the solution. But, if an alteration results in a poorer solution, then another alteration is made to obtain an optimal solution [1].

Algorithm

```
1. PresentNode = startNode;
2. While (not termination) do
3. For j = Neighbors of PresentNode)
4. NextEval = -INF
5. NextNode = NULL
6. For all k in j
7. If (EVAL(x) > nextEval)
8. NextNode = k
9. End if
10. End for
11. End for
12. NextEval = EVAL (k)
13. If nextEval <= EVAL (PresentNode)
14. End if
15. End While
16. //Return present node if no better neighbors exist
17. Return PresentNode
18. PresentNode = nextNode;
```

Hill climbing pseudocode [1]

Application

1. Configure application servers [1]
2. Travelling Salesman's Problem [1]
3. Signature verification [1]

Advantages

4. Uses very little computer resources in its search since it stores only the present solution [1].
5. Returns better results than other algorithms in a situation of unexpected interruption to the algorithms execution [1].

Disadvantages

6. Low speed in instances involving plateaus [1]. A plateau is flat area of the search space in which all neighbouring states have the same value.
7. May fall into local minima in non-convex functions [1]. Local maxima is a state that is better than its neighbours, but may not be better than some other states that are far away.

These disadvantages may be countered using the techniques of backtracking, random-restart hill-climbing, and through moving in several directions at once. **Backtracking** involves moving to some earlier node and to try going in a different direction. **Random-restart** starts hill-climbing searches from randomly generated initial states and running each until it halts or makes no discernible progress. It saves the best result found so far from any of the searches. It can use a fixed number of iterations, or can continue until the best saved result has not been improved for a certain number of iterations.

Simulated Annealing

Kirkpatrick, Gelatt, and Vecchi [24] developed Simulated Annealing (SA) method that models the heating and cooling processes of materials in metallurgical engineering. The algorithm starts with a random search at high temperature and continues decreasing the temperature until it reaches zero. That is, SA is like hill climbing algorithm, except that it allows some downhill moves (to worse states). As the temperature decreases, it gradually allows only small downhill moves [1].

Algorithm

```
1. Let x = x0
2. For y = 0 to ymax :
3. T ← temperature(ky⁄ ymax)
4. Select a neighbor randomly, xnew ← neighbor(x)
5. If P(E(s), E(xnew), T) ≥ random(0, 1), move to the new
state
6. End if
7. End for
8. x ← xnew
9. Output: the final state x
```

Simulated Annealing Pseudocode [1]

Applications

8. Quadratic Assignment Problem [1]
9. Job Shop Scheduling [1]
   Travelling Salesman's Problems [1]
10. N-Queens problem [1]
11. Artificial Neural Networks Training [1]

Advantages

Can escape being trapped in local minima by manipulating the cooling temperature.

Disadvantages

12. Is not very effective in a smooth energy landscape [1].
13. Is not very effective in instances with few local minima [1].
14. Has the problem of delay in arriving at quality solutions [1].

Evolutionary algorithms

Evolutionary algorithms (EAs) [27] are a class of nature inspired algorithms used to find solutions to hard problems, based on the biological evolution in nature. EAs are population-based stochastic search algorithms performing with best-to-survive criteria [2]. Each algorithm commences by creating an initial population of feasible solutions, and evolves iteratively from generation to generation towards a best solution. In successive iterations of the algorithm, fitness-based selection takes place within the population of solutions. Better solutions are preferentially selected for survival into the next generation of solutions [2]. Example of evolutionary algorithms include genetic algorithm (GA) and Differential Evolution (DE) [2].

Genetic Algorithms

Genetic Algorithms (GAs), first proposed by Holland [28], belong to the class of evolutionary algorithms that follow the principles of Charles Darwin Theory of survival of the fittest. GAs begin by initializing a population of solutions. A **population** is a set of individuals, each representing a possible solution (also known as chromosomes) to a given problem. Each **solution/chromosome** is a representation of the problem, usually in the form of a bit vector. Evaluation of **fitness** for each chromosome is done using an appropriate fitness function suitable for the problem. The best chromosome is one that allows best optimization of the fitness function. The best chromosomes, once selected (**selection**), make up the mating pool where they undergo **crossover/recombination** (decomposition and random mixing of two distinct solutions) and **mutation** (randomly perturbing a solution), thus giving new set of solutions (known as **offsprings**). That is, the genetic operators in GA are selection, crossover, and mutation [2]. Genetic algorithms may be classified as **Memetic algorithms** (evolutionary algorithms hybrid with e.g. hill climbing) and **multi-objective genetic algorithms**. It is advantageous to use GAs in the following cases [2]:

15. The search space is large and complex and the domain knowledge is insufficient to encode and/or to narrow the search space.
16. For complex or loosely defined problems.
17. When the traditional search methods fail.

Algorithm

If n is the number of individuals in the population, χ is the fraction of the population to be replaced by crossover in each iteration, and μ is the mutation rate:

```
1. Algorithm: GA(n, χ, μ)[30]
2. // Initialise generation 0:
3. k:= 0;
4. Pk:=a population of n randomly-generated individuals;
5. // EvaluatePk:
6. Computefitness(i)for each i∈Pk;
7. do{
8. // Create generation k+ 1:
9. // 1. Copy:
10.     Select(1−χ)×n members of Pk and insert into Pk+1;
11.     // 2. Crossover:
12.     Select χ×n members of Pk;
13.     pair them up;
14.     produce offspring;
15.     insert the offspring into Pk+1;
16.     // 3. Mutate:
17.     Select μ×n members of Pk+1;
18.     invert a randomly-selected bit in each;
19.     // Evaluate Pk+1:
20.     Computefitness(i)for each I ∈ Pk;
21.     // Increment:
22.     k:=k+ 1;
```

```
    }
    While fitness of fittest individual in Pk is not
  high enough;
    Return the fittest individual from Pk;
```

Applications [2]

26. Optimization problems in data mining and rule extraction
27. Dynamic and multiple criteria web-site optimizations
28. Decision thresholds for distributed detection in wireless sensor networks
29. Computer aided design path planning of mobile robots
30. Fixed charge transportation problem
31. Various scheduling problems
32. Assignment problems
33. Flight control system design
34. Pattern recognition
35. Reactive power dispatch
36. Sensor-based robot path planning
37. Training of radial basis function
38. Multi-objective vehicle routing problem
39. Minimum energy broadcast problem in wireless ad hoc network
40. Software engineering problems
41. Pollutant emission reduction problem in the manufacturing industry
42. Power System Optimization problems
43. Port folio Optimization
44. Optimal learning path in e-learning
45. Web page classification system
46. Closest sting problem in bioinformatics
47. Structural optimization
48. Defect identification system
49. Molecular modelling
50. Web service selection
51. Cutting stock problem
52. Drug design
53. Personalized e-learning system
54. SAT Solvers

Advantages

- GAs have the ability to deal with complex problems: Genetic algorithms can deal with various types of optimization, whether the objective (fitness) function is stationary or nonstationary (changes with time), linear or nonlinear, continuous or discontinuous, or with random noise [31].
- The random mutation guarantees to some extent that we see a wide range of solutions.

- Coding them is easier compared to other algorithms which does the same job.
- Because multiple offsprings in a population act like independent agents, the population (or any subgroup) can explore the search space in many directions simultaneously. This feature makes it ideal to parallelize the algorithms for implementation. Different parameters and even different groups of encoded strings can be manipulated at the same time [31].

Disadvantages [2]

55. May converge towards local optima rather than the global optimum of the problem if the fitness function is not defined properly.
56. Operating on dynamic data sets is difficult.
57. For specific optimization problems, and given the same amount of computation time, simpler optimization algorithms may find better solutions than GAs.
58. Are not directly suitable for solving constraint optimization problems.

Parameters [2]

59. Population size
60. Maximum generation number
61. Crossover probability
62. Mutation probability
63. Length of chromosome
64. Chromosome encoding

Differential Evolution

Another technique in the class of EAs is differential evolution (DE) developed by Price and Storn [8], to optimize real-parameter, real-valued functions. DE is similar to GAs since populations of individuals are used to search for an optimal solution. The main difference between GAs and DE is that, in GAs, mutation is the result of small perturbations to the genes of an individual while in DE mutation is the result of arithmetic combinations of individuals [2].

Algorithm [32]

Algorithm 1. Pseudocode for classic Differential Evolution.
Input: Population size '*NP*', Problem Size 'D', Mutation Rate '*F*', Crossover Rate '*Cr*'; Stope_Criteria {Number of Generation, Target}, Upper Bound 'U', Lower Bound 'L'
Output: Best_Vector
1 Population = Initialize Population (*NP*, D, U, L);
2 While (Stope_Criteria *6*= True) do
3 Best_Vector = EvaluatePopulation (Population);
4 v_x = Select_Random_Vector (Population);
5 Index = FindIndexOfVector (v_x); //specify row number of a vector
6 Select_Random_Vector (Population,v_1, v_2, v_3) where v_1 *6*= v_2 *6*= v_3 *6*= v_x
7 Vy = v_1, + *F*(v_2 - v_3)
8 For (*i* = 0; *i*++; *i* < D - 1)//Loop for starting Crossover operation
9 if (*rand*$_j$ [0, 1) < *CR*)Then
10 u[i] = v_x [i]
11 *Else* u[i] = v_y [i]
12 End For Loop//end crossover operation

13 If (CostFunctionOfVector(u) ≤ CostFunctionOfVector (v_x)) Then
14 UpdatePopulation (u, Index, Population);
15 End; //While loop
16 Return Best_Vector;

Applications [2]

65. Unsupervised image classification
66. Clustering
67. Digital filter design
68. Optimization of non-linear functions
69. Global optimization of non-linear chemical engineering processes
70. Multi-objective optimization

Advantages [2]

71. Easy to implement
72. Requires little parameter tuning
73. Fast convergence
74. Generally reliable, accurate, robust and fast optimization technique

Disadvantages [2]

75. Noise may adversely affect the performance of DE due to its greedy nature.
76. Finding problem-dependent control parameters used in DE is a time consuming task.

A self-adaptive DE (SDE) algorithm can eliminate the need for manual tuning of control parameters.

Parameters [2]

77. Population size
78. Dimension of the problem
79. F scale factor
80. Probability of crossover

Extended Version

81. Memetic algorithms are hybrid evolutionary algorithms combined with other techniques like hill climbing.
82. Multi-objective genetic algorithms allow optimization of multiple objective functions
83. Quantum-inspired algorithms are a class of algorithms that use classical computers to
simulate some physical phenomena such as superposition and entanglement to perform quantum computations. Recent research has focused on exploiting the benefits of quantum computing in the field of evolutionary algorithms [6] and can be divided into three kinds of
algorithms [6]

84. Evolutionary-designed quantum algorithms (EDQA), where the main idea is to use genetic programming to generate new quantum algorithms.
85. Quantum evolutionary algorithms (QEA), which focus on developing algorithms for quantum computers.
86. Quantum-inspired evolutionary algorithms (QIEA), which use concepts of quantum mechanics to develop evolutionary methods for a classical computers. An example is the quantum-inspired *Acromyrmex* evolutionary algorithm (QiAeA), inspired in the colony evolution of the leaf-cutters and weaver ant species. This idea is different to the ant colony optimisation (ACO) where the central idea is to use
swarm intelligence based on food catering to solve complex optimisation problems**.** Although, QiAeA is an ant-inspired method model, but it is different from the current ant-inspired proposals and has a higher similarity to genetic algorithms, where the fittest individuals in a population survive and pass their genes to the next generation [6].

Swarm Intelligence

Eberhart and Kennedy [25] first presented the Swarm Intelligence approach, a recent and emerging paradigm in bio inspired computing for implementing adaptive systems. While EAs are based on **genetic adaptation** of organisms SI is based on collective **social behavior** of organisms. As per definitions in literature, Swarm Intelligent encompasses the implementation of collective intelligence of groups of simple agents that are based on the behaviour of real world insect swarms, as a problem solving tool. The word ―swarm‖ comes from the irregular movements of the particles in the problem space. These trajectory tracking algorithms being inspired by the collective behaviour of animals, exhibit decentralized, self-organized patterns in the foraging process [2].

Applications

- Sentiment Analysis
- In Movies : Graphics in movies like Lord of the Rings trilogy, Troy.
- Unmanned underwater vehicles(UUV):
- Groups of UUVs used as security units
- Only local maps at each UUV
- Joint detection of and attack over enemy vessels by co-ordinating within the group of UUVs Network Routing
- ACO Routing
- Swarm Robotics
- Swarm bots

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), first proposed by Kennedy and Eberhart [26], is a population-based stochastic nature inspired algorithm for effective search optimization. PSO

models the behaviour of, for example, a swarm of birds searching for a food source. It models the velocity and positions of particles in their search for food. The position of each particles in PSO represents a solution to the problem [1].
Each particle in the swarm represents a solution in a high-dimensional space with four vectors, its current position, best position found so far, the best position found by its neighbourhood so far and its velocity and adjusts its position in the search space based on the best position reached by itself and on the best position reached by its neighbourhood during the search process. In each iteration, each particle updates its position and velocity as follows [2]:

$$x_{k+1}^{i} = x_{k}^{i} + v_{k+1}^{i}$$
$$v_{k+1}^{i} = v_{k}^{i} + c_1 r_1 (p_{k}^{i} - x_{k}^{i}) + c_2 r_2 (p_{k}^{g} - x_{k}^{i})$$

where, x_k^i represents particle position;
v_k^i represents particle velocity;
p_k^i represents the best remembered position reached by the particle itself
p_k^g represents the best remembered position reached by the particle's neighbourhood
c_1, c_2 represents cognitive and social parameters
r_1, r_2 are random numbers between 0 and 1

Algorithm

The algorithm starts by initializing the particles in the search space, followed by updating the position and velocity of each particle in each iteration, until it reaches termination condition. Also, the best achieved objective function values for each particle involved in the search is maintained. All the particles converge on the best solution using the information gathered by each particle, the neighbouring particle, and from the entire flock [1].

```
1. Initialize particle
2. Do (Until termination)
3. For individual particle
4. Calculate fitness value
5. If fitness value is better than its overall best fitness
value (pBest) since inception
6. Record the current value as its new pBest
7. End if
8. Select the particle that has the best fitness value in the
swarm as the gBest
9. End for
10. For individual particle
11. Determine the individual particle's velocity using
velocity equation
12. Update the individual particle's position using position
equation until termination condition
13. End for
14. End do
15. Output the best result
```

PSO Algorithm [1]

Applications [1]

87. Complex nonlinear function optimization
88. Communications and control applications
89. Task assignment
90. Antenna design
91. Combinatorial optimization problems
92. Biomedical and pharmaceutical applications
93. Fault diagnosis
94. Travelling Salesman's Problems
95. PID tuning of Automatic Voltage Regulators
96. Global optimization problems
97. Computer networking problems
98. Robotic applications
99. Scheduling
100. Signal processing
101. Human tremor analysis
102. Electric/hybrid vehicle battery pack state of change
103. Human performance assessment
104. Ingredient mix optimization
105. Evolving neural networks to solve Problems

Advantages [1]

- Relatively simple to implement
- Uses relatively fewer parameters than in some algorithms like the Ant colony optimisation.
- More efficient in its use of memory than the Evolutionary algorithms like the GA.
- Allows diversity in its search space since all the particles use the information obtained by the best particles either in the neighbourhood or the overall best in each iteration to improve their positions and speed.

Disadvantages

- Uses several parameters and this affects the algorithm's speed and management of computer resources [1].
- Suffers from the partial optimism, which degrades the regulation of its speed and direction.
- Problems with non coordinate system (for instance, in the energy field)

Parameters [2]

106. Number of particles
107. Dimension of particles
108. Range of particles
109. The particles' velocity
    - Maximum velocity (Vmax) along any dimension
    - If velocity is too low → algorithm too slow
    - If velocity is too high → algorithm too unstable

110. Learning factors:
111. c_1 representing cognitive parameter
112. c_2 representing social parameter
113. Inertia weight
114. Maximum number of iterations

Ant Colony Optimization

Ant colony optimization algorithm, proposed by Dorigo and Di Caro [29], simulates the random movements of ants in search of food. Once the ants discover a food source, they pick particles of the food and on their way back to their nest, deposits some pheromones. The pheromones inform other ants of the food source discovery. Neighbouring ants will then join in bringing food from this food source and will also drop pheromones on their way back. This process increases the pheromone concentration on the shortest route, thereby attracting other ants. As the food exhausts, the pheromone concentration diminishes and the ants stop dropping more pheromones [1].

Algorithm

```
1. Initialize particle
2. Do (Until termination)
3. For individual particle
4. Calculate fitness value
5. If fitness value is better than its overall best fitness value (pBest) since inception
6. Record the current value as its new pBest
7. End if
8. Select the particle that has the best fitness value in the swarm as the gBest
9. End for
10. For individual particle
11. Determine the individual particle's velocity using velocity equation
12. Update the individual particle's position using position equation until termination condition
13. End for
14. End do
15. Output the best result
```

ACO Pseudocode [1]

Applications [1]

- Solving routing problems in computer and telecommunication networks
- Sequential Ordering Problems
- Travelling Salesman's Problems
- Resource Constraint Project Scheduling
- Subset problems
- Machine Learning Problems
- Vehicle routing
- Proportional, Integral and Derivative parameters-tuning of Automatic Voltage Regulators
- Stochastic optimization problems

Advantages [1]

- Flexibility in being hybridized with other heuristics or metaheuristics in a quest for greater efficiency.
- Robust in search situations where the graph is prone to dynamic changes like new distances.
- Performs well in distributed computing environments
- Exhibits inherent parallelism

Disadvantages

- Premature convergence [1]
- Uses several parameters that require proper tuning and also affect the algorithm speed [1]
- Theoretical analysis is difficult
- Sequences of random decisions (not independent)
- Research is experimental rather than theoretical

Parameters [1]

- Number of ants
- Number of iterations ,
- Pheromone quantity
- Pheromone update rule
- Evaporation rate
- Pheromone reinforcement rate

Extended Version

Various extensions of PSO and ACO have been proposed in the literature including parallel implementations, extensions for big data, hybridizations, multi-objective optimizations, and ensemble approaches [4, 7].

Parallel clustered PSO algorithm (PCPSO) based approach **for big data-driven** service composition that handles huge amounts of heterogeneous data and processes data using parallel processing with MapReduce in the Hadoop platform [4]. Cat swarm optimization (CSO) algorithm-based approach for big data classification to choose characteristics during classification of text for big data analytics. The CSO algorithm uses the term frequency-inverse document occurrence to improve accuracy of feature selection [4]. Swarm intelligence (SI) algorithm-based big data analytics approach for the economic load dispatch problem and the SI algorithm handles the high dimensional data, which improves the accuracy of the data processing [4]. Multispecies optimizer (PS2O) algorithm-based approach for data stream mining big data to select features. An incremental classification algorithm is used in the PS2O algorithm to classify the collected data streams pertaining to big data, which enhanced the analytical accuracy within a reasonable processing time [4]. Ant colony optimization (ACO) algorithm-based approach for mobile big data using rough set. The ACO algorithm helps to select an optimal feature for resolved decisions, which aids in effectively managing big data from social networks (tweets and posts) [4]. The improved ACO algorithm (IACO)- based big data analytical approach for management of medical data such as patient data, operation data etc., which helps doctors retrieve the required data quickly [4].

Recent years have seen the proposal of **hybridized PSO** implementations that integrates the inherent social, co-operative character of the algorithm with tested optimization strategies arising out of distinctly different traditional or evolutionary paradigms towards achieving the central goal of intelligent exploration-exploitation. This is particularly helpful in offsetting weaknesses in the underlying algorithms and distributing the randomness in a guided way [7]. For example, popular approaches in hybridizing GA and PSO involve using the two approaches sequentially or in parallel or by using GA operators such as selection, mutation and reproduction within the PSO framework [7]. Examples include GA-PSO [9] algorithm for Network Design; GAI-PSO [10] for Data Clustering; Life Cycle Model [11], Breeding Swarm [12], PSO-RDL [13], HPSOM [14], GA-PSO [9], GSO [15] for Unconstrained Global Optimization; and DPSO-mutation-crossover [16] for Document Clustering, to name a few. Also, several approaches of hybridizing DE with PSO exist in the literature [7] including DEPSO-Scout [17] for Numerical Optimization; Multi-DEPSO [18] for Dynamic Optimization, to name a few. On similar lines, hybridization of PSO using Simulated Annealing (SA), Ant Colony Optimization (ACO), and other Metaheuristic Approaches can be found in the literature [7].
The No Free Lunch (NFL) Theorem [20] by Wolpert and Macready establishes that no single optimization algorithm can produce superior results when averaged over all objective functions. Instead, different algorithms perform with different degrees of effectiveness given an optimization problem. To this end, researchers have tried to put together **ensembles of optimizers** to obtain a set of candidate solutions given an objective function and choose from the promising ones. Existing ensemble approaches include the Multi-Strategy Ensemble PSO (MEPSO) [19], the Heterogenous PSO [21], the Ensemble Particle Swarm Optimizer [22] by Lynn and Suganthan [7].

Artificial neural networks (ANN)

The inventor of the first neurocomputer, Dr. Robert Hecht-Nielsen, defines a neural network as − "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs." [33]. The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living **neurons** and **dendrites**. The human brain is composed of 86 billion nerve cells called **neurons.** They are connected to other thousand cells by **Axons.** Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward [33].

ANNs are composed of multiple **nodes**, which imitate biological **neurons** of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its **activation** or **node value.** Each link is associated with **weight.** ANNs are capable of learning, which takes place by altering weight values. Figure 2 shows a simple ANN [33].

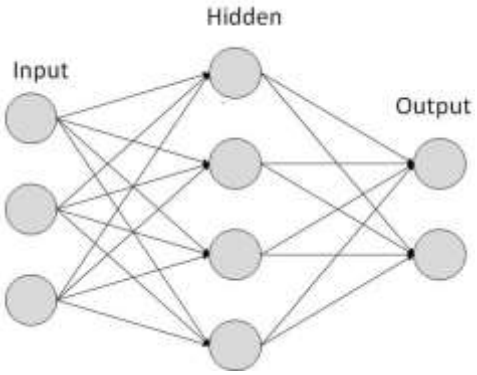


Figure 2 A simple ANN

How does a simple neuron works ? Let there are two neurons **X** and **Y** which is transmitting signal to another neuron **Z** . Then, **X** and **Y** are input neurons for transmitting signals and **Z** is output neuron for receiving signal. The input neurons are connected to the output neuron over an interconnection links (**A** and **B**) (figure 2a). For the ANN in the figure 2a, the net input has to be calculated in the following way [34]:

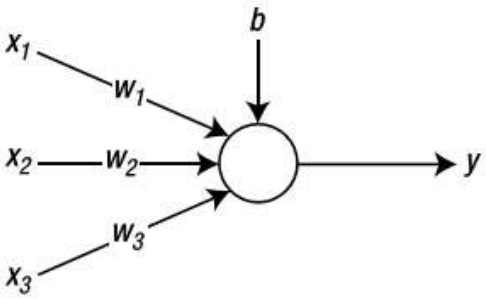


Figure 3a An example ANN [34]

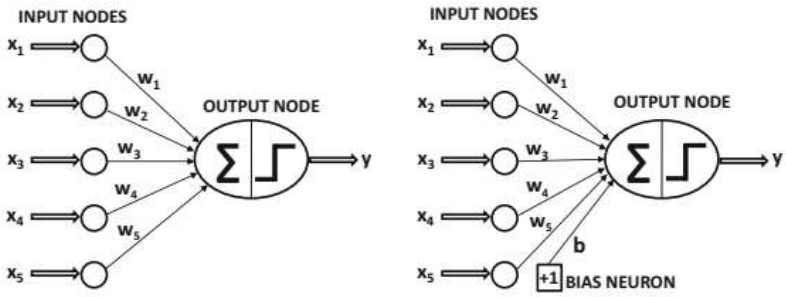
Given (figure 2a) that x_1, x_2, $and\ x_3$ are the input signals; w_1, w_2, $and\ w_3$ are the weights for the corresponding signals; b is the bias, which is a factor associated with the storage of information, the weighted sum of the input signals is computed as: $v = wx + b$

where, $w = [w_1\ w_2\ w_3]$, $x = [x_1\ x_2\ x_3]'$

The output from the activation function (φ) from the weighted sum is given by: $y = \varphi(v)$.

The Basic Architecture of Neural Networks

Neural networks may be classified as single-layer and multi-layer neural networks. In the **single-layer network**, a set of inputs is directly mapped to an output by using a generalized variation of a linear function. This simple instantiation of a neural network is also referred to as the **perceptron,** proposed by Rosenblatt [36].



(a) Perceptron without bias (b) Perceptron with bias

Figure 3 Architecture of a perceptron[35]

The simplest neural network is referred to as the perceptron (figure 3). This neural network contains a single input layer and an output node. Given that each training instance is of the form (X, y), where each $X = [x1, \ldots xd]$ contains d feature variables, and $y \in \{-1, +1\}$ contains the *observed value* of the binary class variable. The goal is to predict the class variable for cases in which it is not observed. The input layer contains d nodes that transmit the d features $X = [x1 \ldots xd]$ with edges of weight $W = [w1 \ldots wd]$ to an output node. The input layer does not perform any computation in its own right. The linear function $\sum w_i x_i$ (for perceptron without bias), $\sum w_i x_i + b$ (for perceptron with bias) is computed at the output node. Subsequently, the sign of this real value is used in order to predict the dependent variable of X. Therefore, the prediction $\hat{}\ y$ is computed as follows: $y = sign\{\sum w_i x_i\}$ (for perceptron without bias), $y = sign\{\sum w_i x_i + b\}$ (for perceptron with bias). The sign function maps a real value to either +1 or -1 [35]. In **multi-layer** neural networks, the neurons are arranged in layered fashion, in which the input and output layers are separated by a group of hidden layers. This layer-wise architecture of the neural network is also referred to as a **feed-forward** network (figure 3) [35]. In a feedforward ANN, the information flow is unidirectional. A unit sends information to other unit from which it does not receive any information. There are no feedback loops. They are used in pattern generation/recognition/classification. They have fixed inputs and outputs [33]. However, in **feedback ANN** (figure 4) [33], feedback loops are allowed.

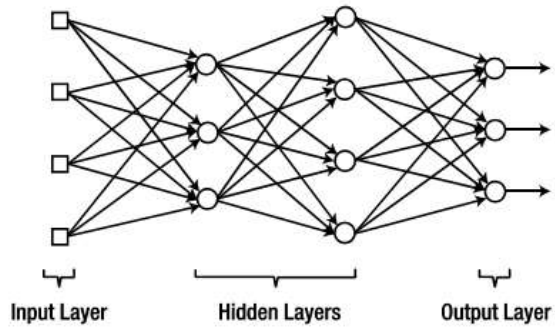


Figure 4 Example feedforward networks [33].

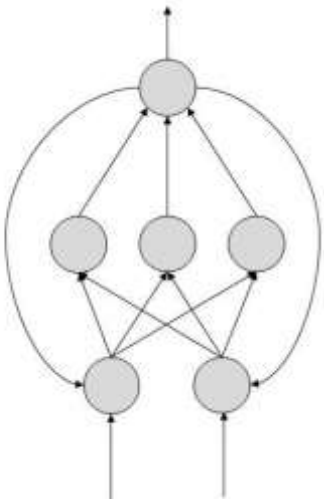


Figure 5 Example feedback network [33].

Machine Learning in ANNs: ANNs are capable of learning and they need to be trained. There are several learning strategies [33]−

- **Supervised Learning** − It involves a teacher that is scholar than the ANN itself. For example, the teacher feeds some example data about which the teacher already knows the answers. For example, pattern recognizing. The ANN comes up with guesses while recognizing. Then the teacher provides the ANN with the answers. The network then compares it guesses with the teacher's "correct" answers and makes adjustments according to errors [33].
- **Unsupervised Learning** − It is required when there is no example data set with known answers. For example, searching for a hidden pattern. In this case, clustering i.e. dividing a set of elements into groups according to some unknown pattern is carried out based on the existing data sets present [33].
- **Reinforcement Learning** − This strategy built on observation. The ANN makes a decision by observing its environment. If the observation is negative, the network adjusts its weights to be able to make a different required decision the next time [33].

Algorithm

The Perceptron Algorithm (figure 3)

1. For every input, multiply that input by its weight.
2. Sum all of the weighted inputs.
3. Compute the output of the perceptron based on that sum passed through an activation function (the sign of the sum)

Algorithm for Forward Propagation (figure 4):

Input nodes i, given input x_i:

For each input node I, $output_i = x_i$

Hidden layer nodes j:

For each hidden neuron j, $output_j = \sum \varphi(w_{ji} output_i)$

Output layer neurons k:

For each output neuron k, $output_k = \sum \varphi(w_{kj} output_j)$

Back Propagation Algorithm

In the single-layer neural network, the training process is relatively straightforward because the error (or loss function) can be computed as a direct function of the weight. In the case of multi-layer networks, the problem is that the loss is a complicated composition function of the weights in earlier layers. Thus there is a requirement of the backpropagation algorithm. [35]. Backpropagation is the training or learning algorithm that learns by example. Depending on the submitted training data, the algorithm changes the network's weights so that it can produce desired output for a particular input on finishing the training. Back Propagation networks are ideal for simple Pattern Recognition and Mapping Tasks [33].

Algorithm for Backpropagation

1. Initialize the weights
2. Input the given training data into the neural network
3. Calculate the error based on the correct output
4. Reduce the error by adjusting the weights
5. Repeat the steps 2-4 for the entire training data

Algorithm for training of a Single-Layer Neural Network using the Delta Rule

For a single layer network, let d_i be the correct output of the output node i. The weight is changed by an amount proportional to the difference between the desired output (d_i) and the actual output (y_i).

1. Initialize the weights
2. Input the given training data into the neural network
3. Calculate the error (e_i) based on the correct output: $e_i = d_i - y_i$
4. Calculate the weight adjustment according to delta rule: $\Delta w_{ij} = \alpha\, e_i\, x_j$
5. Reduce the error by adjusting the weights by: $w_{ij} = w_{ij} + w_{ij}$
6. Repeat the steps 2-4 for the entire training data
7. Repeat the steps 2-5 till the error becomes acceptable.

Applications

They can perform tasks that are easy for a human but difficult for a machine [33]−

- **Aerospace** − Autopilot aircrafts, aircraft fault detection.
- **Automotive** − Automobile guidance systems.
- **Military** − Weapon orientation and steering, target tracking, object discrimination, facial recognition, signal/image identification.
- **Electronics** − Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.

- **Financial** − Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis, currency value prediction, document readers, credit application evaluators.
- **Industrial** − Manufacturing process control, product design and analysis, quality inspection systems, welding quality analysis, paper quality prediction, chemical product design analysis, dynamic modeling of chemical process systems, machine maintenance analysis, project bidding, planning, and management.
- **Medical** − Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.
- **Speech** − Speech recognition, speech classification, text to speech conversion.
- **Telecommunications** − Image and data compression, automated information services, real-time spoken language translation.
- **Transportation** − Truck Brake system diagnosis, vehicle scheduling, routing systems.
- **Software** − Pattern Recognition in facial recognition, optical character recognition, etc.
- **Time Series Prediction** − ANNs are used to make predictions on stocks and natural calamities.
- **Signal Processing** − Neural networks can be trained to process an audio signal and filter it appropriately in the hearing aids.
- **Control** − ANNs are often used to make steering decisions of physical vehicles.
- **Anomaly Detection** − As ANNs are expert at recognizing patterns, they can also be trained to generate an output when something unusual occurs that misfits the pattern.

Advantages [37]

- Storing information on the entire network: Information such as in traditional programming is stored on the entire network, not on a database. The disappearance of a few pieces of information in one place does not restrict the network from functioning.
- The ability to work with inadequate knowledge: After ANN training, the data may produce output even with incomplete information. The lack of performance here depends on the importance of the missing information.
- It has fault tolerance: Corruption of one or more cells of ANN does not prevent it from generating output. This feature makes the networks fault-tolerant.
- Having a distributed memory: For ANN to be able to learn, it is necessary to determine the examples and to teach the network according to the desired output by showing these examples to the network. The network's progress is directly proportional to the selected instances, and if the event can not be shown to the network in all its aspects, the network can produce incorrect output
- Gradual corruption: A network slows over time and undergoes relative degradation. The network problem does not immediately corrode.
- Ability to train machine: Artificial neural networks learn events and make decisions by commenting on similar events.
- Parallel processing ability: Artificial neural networks have numerical strength that can perform more than one job at the same time.

Disadvantages [37]

- Hardware dependence: Artificial neural networks require processors with parallel processing power, by their structure. For this reason, the realization of the equipment is dependent.
- Unexplained functioning of the network: This is the most important problem of ANN. When ANN gives a probing solution, it does not give a clue as to why and how. This reduces trust in the network.
- Assurance of proper network structure: There is no specific rule for determining the structure of artificial neural networks. The appropriate network structure is achieved through experience and trial and error.
- The difficulty of showing the problem to the network: ANNs can work with numerical information. Problems have to be translated into numerical values before being introduced to ANN. The display mechanism to be determined here will directly influence the performance of the network. This depends on the user's ability.
- The duration of the network is unknown: The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.

Parameters

- Number of layers
- Number of neurons per layer
- Number of training iterations,
- Number of hidden neurons
- Learning rate
- Momentum parameter: Used to prevent the system from converging to a local minimum or saddle point.

Extended Version

Deep Neural Networks (DNNs) (figure 6), also called convolutional networks, are composed of multiple levels of nonlinear operations, such as neural nets with many hidden layers. Deep learning methods aim at learning feature hierarchies, where features at higher levels of the hierarchy are formed using the features at lower levels. Much better results are achieved in deeper architectures when each layer is pre-trained with an unsupervised learning algorithm. Then the network is trained in a supervised mode using back-propagation algorithm to adjust weights [38].

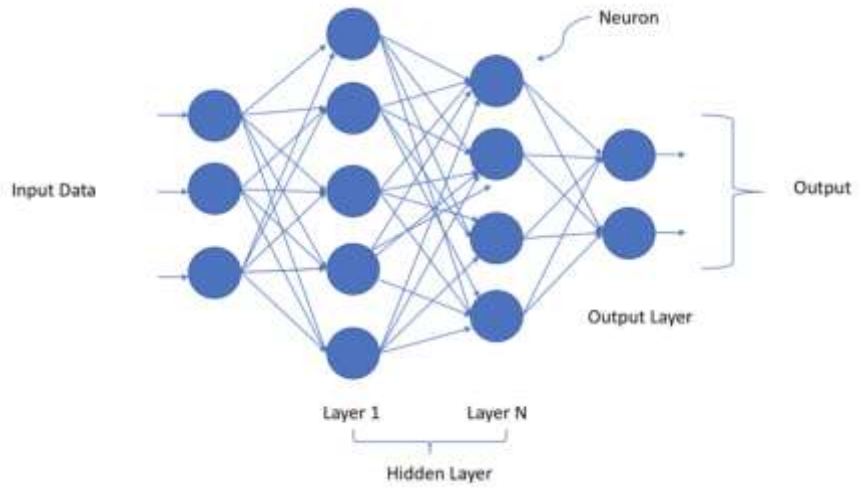


Figure 6 Deep Neural network with n layers [39]

Future Work possible

One of the future directions is combining the strengths of **population-based algorithms and big data analytics** to design new algorithms on the optimization or data analytics. The big data is created in many areas in our everyday life, for example, in internet data mining. That is, the big data problem could be analyzed from the perspective of computational intelligence and meta-heuristic global optimization [5].

Another future direction is towards the hybridization of existing algorithms [7].

Some pressing issues which are listed below merit further work by the PSO community [7]. 1. Parameter Sensitivity: Solution quality of metaheuristics like PSO are sensitive to their parametric evolutions. This means that the same strategy of parameter selection does not work for every problem.

2. Convergence to local optima: Unless the basic PSO is substantially modified to take into account the modalities of the objective function, more often than not it falls prey to local optima in the search space for sufficiently complex objective functions.

3. Subpar performance in multi-objective optimization for high dimensional problems: Although niching techniques render acceptable solutions for multimodal functions in both static and dynamic environments, the solution quality falls sharply when the dimensionality of the problem increases. Ensemble optimizers, although promising, do not address the underlying shortcomings of the basic PSO. Theoretical issues such as the particle explosion problem, loss of particle diversity as well as stagnation to local optima deserve the attention of researchers so that a unified algorithmic framework with more intelligent self-adaptation and less user-specified customizations can be realized for future applications.

Bibliography

1. ODILI, J. B., NORAZIAH, A., AMBAR, R., & WAHAB, M. H. A. A Critical Review of Major Nature-Inspired Optimization Algorithms. *The Eurasia Proceedings of Science Technology Engineering and Mathematics*, (2), 376-394.
2. Binitha, S., & Sathya, S. S. (2012). A survey of bio inspired optimization algorithms. *International journal of soft computing and engineering*, *2*(2), 137-151.
3. Mishra, A. (2017). Nature inspired algorithms: a survey of the state of the art. *International Journal*, *5*(9), 16-21.
4. Gill, S. S., & Buyya, R. (2019). Bio-inspired algorithms for big data analytics: a survey, taxonomy, and open challenges. In *Big Data Analytics for Intelligent Healthcare Management* (pp. 1-17). Academic Press.
5. Cheng, S., Liu, B., Ting, T. O., Qin, Q., Shi, Y., & Huang, K. (2016). Survey on data science with population-based algorithms. *Big Data Analytics*, *1*(1), 3.
6. Montiel, O., Rubio, Y., Olvera, C., & Rivera, A. (2019). Quantum-Inspired Acromyrmex Evolutionary Algorithm. *Scientific reports*, *9*(1), 1-10.
7. Sengupta, S., Basak, S., & Peters, R. A. (2019). Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, *1*(1), 157-191.
8. Price, K., Storn, R. M., & Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.
9. Juang, C. F. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*(2), 997-1006.
10. Abdel-Kader, R. F. (2010, February). Genetically improved PSO algorithm for efficient data clustering. In *2010 Second International Conference on Machine Learning and Computing* (pp. 71-75). IEEE.
11. Krink, T., & Løvbjerg, M. (2002, September). The lifecycle model: combining particle swarm optimisation, genetic algorithms and hillclimbers. In *International Conference on Parallel Problem Solving from Nature* (pp. 621-630). Springer, Berlin, Heidelberg.
12. Settles, M., & Soule, T. (2005, June). Breeding swarms: a GA/PSO hybrid. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation* (pp. 161-168).
13. Jian, M. C., & Chen, Y. P. (2006, July). Introducing recombination with dynamic linkage discovery to particle swarm optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 85-86).
14. Esmin, A. A. A., Lambert-Torres, G., & Alvarenga, G. B. (2006, December). Hybrid evolutionary algorithm based on PSO and GA mutation. In *2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06)* (pp. 57-57). IEEE.
15. Grimaldi, E. A., Grimaccia, F., Mussetta, M., Pirinoli, P., & Zich, R. F. (2004, November). A new hybrid genetical-swarm algorithm for electromagnetic

optimization. In *Proceedings. ICCEA 2004. 2004 3rd International Conference on Computational Electromagnetics and Its Applications, 2004.* (pp. 157-160). IEEE.

16. Premalatha, K., & Natarajan, A. M. (2009). Discrete PSO with GA operators for document clustering. *International Journal of Recent Trends in Engineering*, *1*(1), 20.

17. Boonserm, P., & Sitjongsataporn, S. (2017, March). A robust and efficient algorithm for numerical optimization problem: DEPSO-Scout: A new hybrid algorithm based on DEPSO and ABC. In *2017 International Electrical Engineering Congress (iEECON)* (pp. 1-4). IEEE.

18. Xiao, L., & Zuo, X. (2012, June). Multi-DEPSO: A DE and PSO based hybrid algorithm in dynamic environments. In *2012 IEEE Congress on Evolutionary Computation* (pp. 1-7). IEEE.

19. Du, W., & Li, B. (2008). Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information sciences*, *178*(15), 3096-3109.

20. Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82.

21. Engelbrecht, A. P. (2010, September). Heterogeneous particle swarm optimization. In *International Conference on Swarm Intelligence* (pp. 191-202). Springer, Berlin, Heidelberg.

22. Lynn, N., & Suganthan, P. N. (2017). Ensemble particle swarm optimizer. *Applied Soft Computing*, *55*, 533-548.

23. Selman, B., & Gomes, C. P. (2006). Hill-climbing Search. *Encyclopedia of cognitive science*.

24. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

25. Eberhart, R. C., Shi, Y., & Kennedy, J. (2001). *Swarm intelligence*. Elsevier.

26. Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks* (Vol. 4, pp. 1942-1948). IEEE.

27. Back, T. (1996). Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press.

28. Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, *2*(2), 88-105.

29. Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *26*(1), 29-41.

30. **http://www.cs.ucc.ie/~dgb/courses/tai/notes/handout12.pdf**

31. **https://www.sciencedirect.com/topics/computer-science/genetic-algorithm**

32. Eltaeib, T., & Mahmood, A. (2018). Differential evolution: A survey and analysis. *Applied Sciences*, *8*(10), 1945.

33. **https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm**

34. **https://www.geeksforgeeks.org/introduction-artificial-neural-network-set-2/**

35. Aggarwal, C. C. (2018). Neural networks and deep learning. *Springer*, *10*, 978-3.
36. Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.
37. https://intellipaat.com/community/21886/advantages-and-disadvantages-of-neural-networks
38. https://www.sciencedirect.com/topics/computer-science/deep-neural-network
39. https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb