The Implication of Deep Neural Networks in Solving Optimization Problems for Network Security

Shabbir Hassan
Department of Computer Science
Aligarh Muslim University
Aligarh, 202002, India

ABSTRACT
Optimization which implies minimization and maximization of some objective functions often becomes heuristics, as all the problems are not just in the form of linear or polynomial. To optimize problems we may apply heuristics method or any other type of approximation method that can be employed. On the application of derivatives and partial derivatives, these evolutionary algorithms liberalize the objective functions and their restrictions at a specific point. The objective function approximation method of (NLO) Non-linear optimization which used to resolve the optimization problems efficiently. This study paper proposes the critical use of artificial neural networks to strategically optimize these problems so that to apply other possible techniques or methods if it could not be optimized directly. We have enforced the conversion of problems into polynomials so that the solution of Optimization problems (OP) can be calculated accurately.

Keywords
Deep leering neural network (DNN), Neural network model, Optimization problems (OP), Non-linear optimization (NLO), Particle Swarm Optimization (PSO), Method of Approximation (MAP), Unprotected AES implementation.

General Terms
Stream optimization algorithms, curve estimation function. Recognition, pre-trained multilayer perception.

1. INTRODUCTION
DNN is some part of soft computing and has applied to different fields so that as intelligence, industry, logistics, information theory or control system [1, 3]. The best techniques to get an optimized solution are by involving objective maximizing and minimizing functions and some linear limitations. Optimization problems play a crucial role in all kinds of fields regarding DNN and AI. In the case of non-linear restrictions heuristics are used to obtain a pseudo optimal solution. Linear restrictions are easily solvable by Simplex [1, 2, 6] algorithms. This restricts the deeper study of these kinds of problems. In many cases, Lagrange Multipliers or Kuhn-Tucker multipliers can't solve all the problems in most numerical terms, so in such kind of cases used heuristics and met heuristics, the solution comes into the picture such as genetic algorithms PSO [8, 9] Simulated annealing ant colony optimization. Here we go for the usage of neural networks such as heuristics when Lagrange multipliers don't make the work fruitful. In the case of objective functions, we use a multilayer perception as the way we implement it on restrictions of optimization. For some defined variables we check required conditions to conduct the training, the activation function to be used is established through the work. This process could result in a transformed objective function that can be resolved without using meta-heuristics. Thus the objective function resulted can be approximated with non-linear regression which gives us a way to solve the problem incurred [27, 28, 29]. We choose the activation function in a way such that after its application on the objective function it results in a form that depicts a polynomial when derivation is made. Once we have the new objective functions in our hands it's easy to calculate the problem in other ways. Non-equality restrictions can be made easy in this manner but we need to make use of gaps to satisfy the restrictions [31].

2. APPLICATION OF HEURISTICS IN OPTIMIZATION
In some cases, optimization cannot be done using simplex or Lagrange tests because simplex texts are used only when the problem is linear whereas Lagrange solves when the problem is non-linear, but it is not true in all the cases so when the algorithm does not follow the optimal solution we use heuristics and Met heuristics. Heuristics such as ant colony optimization uses graphs [21, 12] although intelligence [30, 33], industry [16, 12], logistics [2, 3] information theory [4] or control system [5] to solve problems. Few heuristic methods work on approximation functions, in mathematics. The approximation functions are defined on a certain point which makes it feasible to work on polynomials using Taylor's theorem. On working with this idea we can even solve non-linear optimization problems using Taylor's non-linear functions. This idea is applied in algorithms like Frank-Wolfe [3, 15] in this, we derivate a certain problem on a point calculating straight lines, planes, and hyperplane crosses through that point. The solutions are calculated iteratively using a new hyperplane over each iteration. MAP is the generalized version of the Frank Wolfe algorithm permitting the linearization of restrictions. The hectic work of calculating a new approximation for each tentative solution alternatively can be made by the simple use of neural networks.

3. RELATED WORK
Deep Belief Networks (DBN) has been introduced by Anna L. Buczak et al [41]. It is a class of DNNs made up of multiple layers of hidden units that link the layers, but not each layer. DBNs are uncontrollably educated. They are normally trained individually to recreate the inputs by changing weights in each hidden layer. The Autoencoders are a type of neural network that is unconsidered and in which the network takes a vector as its input and tries to adapt the output to the same vector. By entering, changing the dimension and restoring the input, a higher or lower dimension of data can be generated. Such neural network types are incredibly flexible as they know unattended compressed data encoding. Besides, the computational resources required to construct an efficient model can be trained one layer at a time [7]. The Network is used for encoding the data if the bulky layers are of a smaller dimension than the input and output layers [8]. A noise

remover can be rendered by an autoencoder to re-engineer the input from a noisy input version (refer to Fig 4), called a debruising autoencoder [19] and become stronger by training an autoencoder. It has been demonstrated that this technique is more common and reliable than traditional auto-encoders.

In the Literature of machine-learning applications, Yalin E. Sagduyu et al [42] have addressed safety problems for IoT systems. In order to first differentiate between the traffic generated by IoT and non-IoT devices and then decide the IoT device class, a machine learning multistage Meta classification was equipped. In order to automatically identify device types linked to the IoT network and to implement rules to limit the contact of compromised devices to mitigate the harm resulting from their compromise, machine learning has been employed. Deep learning has been used to detect IoT devices' data injection and screening. In [20] machine learning was applied to detect errors in data sent from edge equipment within an IoT gateway. The leaning of an opponent engine has started to identify wireless communications applications. By changing the input data to a modulation classifier based on profound learning the white box and black box attacks were planned. Our methodology varies as it focuses on transmission and spectrum measurement processes and explains specifically how to input data should be treated in research and training processes. Deep education was used to develop jamming attacks and protection mechanisms, while a spectrum poisoning attack was studied but retraining was not included in these studies.

Ho Bae et al. [43] has developed data-driven systems for numerous practical applications, including the use of vast volumes of medical prediction data for health care network logs for standalone device security assessments [16], and visually impaired car driving [15]. However, in significant number of literature, vulnerabilities in DL systems have recently been found. Such implementations may be dangerous because they are based on a restricted understanding of the DL systems' protection and privacy.

4. RESEARCH GAP

There are several limitations and problems associated with the above-said area are:

4.1 Global optimality conditions for DNN

Several errors field of deep linear and nonlinear neural networks. Minimizing the loss of a deep linear neural network is a non-convex problem, and our understanding of this surface of loss is still incomplete despite recent advances [13]. To be a global minimum, we pose necessary and adequate conditions for deep linear networks to be a critical point of the risk function. Surprisingly, our conditions have a globally optimal test that can be easily tested, whereas those tests are usually intractable in non-convex optimization [32].

4.2 Solving inverse problems with DNN

A core task in various scientific fields is to recover a function or high-dimensional parameter vector from indirect measurements. Recently, new algorithms have emerged that use deep learning and neural networks for inverse problems [11]. These techniques, though still in their infancy, show astounding problems for applications such as low-dose CTs or various sparse data. However, in inverse problems, there are few theoretical findings for profound research [33].

4.3 Risk versus uncertainty in DL

There is an important difference between risk and uncertainty in sequential decision problems. The demarcation between risk and uncertainty is related to the particular model type, in this case, a Bernoulli random variable; even the outcome of a coin might not be risky at all with a more detailed model of flip dynamics. Our distinction is that unlike risk, uncertainty captures the volatility of the posterior belief of an agent that can be resolved by statistical analysis of the relevant data [10]. This distinction reflects a key dichotomy for a learning agent looking to optimize accumulated utility over time [34, 35]. A result of linear regression with a little and a large amount of data is in Fig 1 and 2 below:

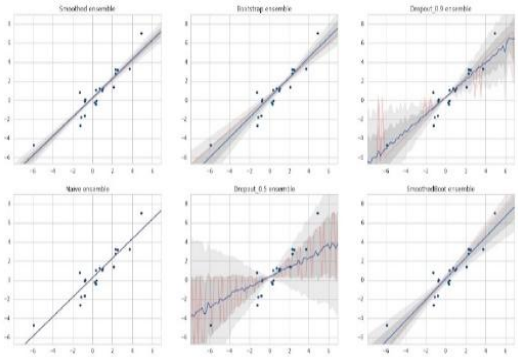

Fig 1: Linear regression with insufficient evidence of data

And here is the simulation report of linear regression with sufficient loss of data

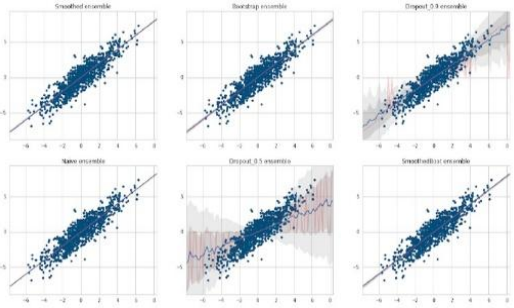

Fig 2: Linear regression with sufficient loss of data

4.4 Low confidence predictions for UI

Recently, deep neural networks (DNNs) have achieved state-of-the-art success on various pattern-recognition tasks or an unrecognizable image (UI), most notably visual classification issues. Since DNNs are now able to classify objects into near-human-level performance images, questions naturally arise as to what disparities between machine and human vision still exist. A recent study revealed that altering an image (e.g. a lion) in a manner that is imperceptible to humans can cause a DNN to mark the picture entirely as something else such as mislabeling a library lion [35]. It is possible to create unrecognizable images for human eyes that DNNs almost certainly believe to be familiar objects, which we call "fooling pictures" more generally, fooling examples. Our findings shed light on fascinating variations between human vision and current DNNs and raise concerns regarding DNN machine vision in general [36].

4.5 Problems in safety verification of DNN

Deep neural networks have produced remarkable experimental results in the classification of images, but can be surprisingly unstable concerning adversarial perturbations, i.e. small changes in the input image that cause the network to misclassify it [29].

4.6 Unsolved ill-posed inverse problems

Use iterative deep neural networks to solve ill-positioned inverse problems. The approach results in a gradient-like iterative scheme where the function "gradient" is learned in each iteration using a co-evolutionary network that includes the gradients of the data discrepancy and the regularize as input [37].

5. CRYPTOGRAPHIC APPLICATIONS OF DNN

The question we are discussing is the following: how can a customer use a third-party predictive model without sacrificing private information. A hospital, for example, may want to use a cloud service to predict a patient's risk of readmission. Nonetheless, the patient's medical files cannot be released due to legislation [14, 45]. The goal is to make an inference using the model, without jeopardizing the prediction accuracy or data privacy. We use neural networks to achieve high precision which has been shown to outperform other learning models for many tasks. To satisfy the privacy requirements, in the following protocol we use homomorphic encryption: the data owner encrypts the data and sends the ciphertexts to the third party to obtain a prediction from a qualified model. The code works on such ciphertexts and sends back the prediction that is encrypted. Not only the data remain private in this protocol, but even the expected values are accessible only to the data owner [39]. Using homomorphic encryption and modifications to neural network activation functions and training algorithms, we demonstrate that protocol is possible and maybe feasible. This approach paves the way for the development of stable neural network prediction services based on the cloud without violating the privacy of users [28, 46].

The most popular and effective profiled side-channel attack is a template attack. It relies on a rational assumption about the noise of the system under attack: a multivariate Gaussian distribution is the probability density function of the results. To ease this presumption, a recent line of research has been exploring new approaches to profiling mainly through the application of machine learning techniques. Compared with template attack, the results obtained are commensurate, and in some specific cases stronger. In this work, we propose to continue this recent research line through the application of more sophisticated profiling techniques focused on profound learning [38, 47].

Another problem is whether neural networks can learn to shield knowledge from other neural networks using hidden keys. Specifically, in a multi-agent scheme, we concentrate on preserving confidentiality properties and define those properties as an adversary. Thus, a framework that consists of neural networks called Alice and Bob, and we intend to restrict what a third neural network called Eve learns from eavesdropping about Alice and Bob's communication. Such neural networks are not recommended for unique cryptographic algorithms; rather, we practice end-to-end, adversarial [40, 44].

6. THE PROPOSED IDEA

Any continuous function can be precisely defined as Kolmogorov's theorem using a multilayer perception with a three-layer. However, for the approximation to be exact, we define the activation function and parameters. It is not possible to apply any activation function just as it is because the objective of the function needs to be accounted for. A dataset is generated that trains neural networks in the domain of variables. A multilayer perception is used to regenerate the objective function of the optimization problem. Hence generating a polynomial equation. At last, the objective function is calculated forming a new solution that is therefore applied to resolve a problem. Finally, another solution is applied to resolve the new objective function calculated [17, 25]. The definition of neural networks must include parameters such as the connections, number of layers, activation functions, propagation rules, etc. There are two stages in a multilayer perception that is the learning stage and the prediction process [18, 30]. The layers number and activation functions are the same in both cases. Learning rate and momentum are irrelevant in prediction stage Propagation rule is the weighted sum in case of multilayer perception which is defined as in equation (1). Fig 3 represents the working principle of LOP in neural networks [29]

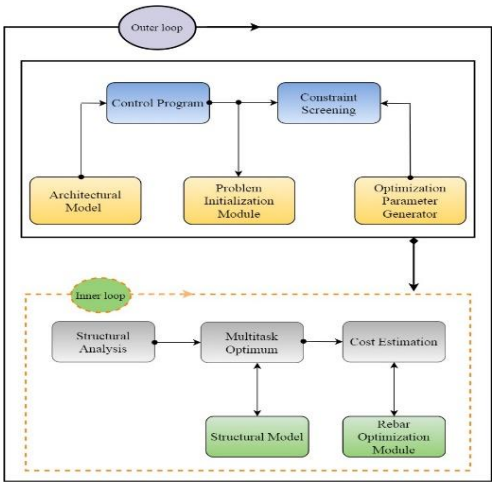

Fig 3: Workflow of linear optimization problems

$$\sum_{j=1}^{n} p \ \forall \ p = w_{j,k} y_i(t) \qquad (1)$$

Where weight $w_{j,k}$ is the foundation of DNN connects neuron $'\boldsymbol{j}'$ in weight input layer allow with artificial neuron $'\boldsymbol{k}'$ in weight hidden layer, y_i is the weight output layer with artificial neuron $'\boldsymbol{j}'$ in the weight input layer, $'\boldsymbol{n}'$ is number artificial neuron of weight input layer $'\boldsymbol{j}'$ at a time $'\boldsymbol{t}'$ [18, 26]. In the case of having an artificial bias neuron, what would result in equation (2) [20, 23]?

$$\sum_{j=1}^{n} p + \Theta(\mathrm{k}) \qquad (2)$$

If the result from activating function is linear then the neuron $'\boldsymbol{k}'$ is combined in linear with neurons of the input layer, and $y_1(k)$ as a linear function then, the present answer corresponds to the answer of a neuron when the activation function is identical [19, 20, 24]. When $'\boldsymbol{M}'$ is in the output layer has the activating functionf_1, the solution is written as an equation (3).

$$y_1\mathrm{k(t)} = f_1\left(\sum_{j=1}^{n} p + \Theta(\mathrm{k})\right) \qquad (3)$$

We should apply the activation function after calculating the propagation rules. The multilayer perception has three layers [21, 33]. We need to apply the propagation rule on two cases to transmit the values from input layer neurons to output layer neurons.

$$y_1\mathrm{M(t)} = \sum_{j=1}^{m_1} w_k(M) \cdot y_j(\mathrm{t}) + \Theta(\mathrm{M}) \qquad (4)$$

Where artificial neuron $'\boldsymbol{M}'$ represents in the weight output layer, and $\boldsymbol{m_1}$ represents many hidden layer of artificial neurons [26, 29, 34].

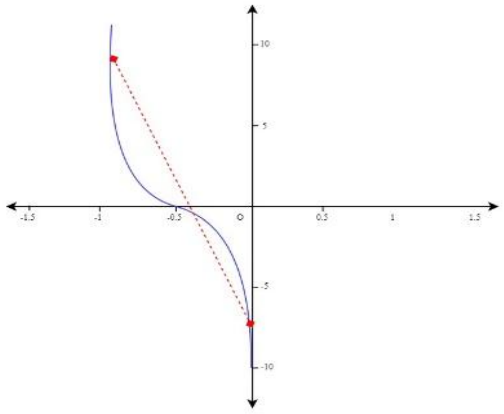

Fig 4 (a): Sigmoid function

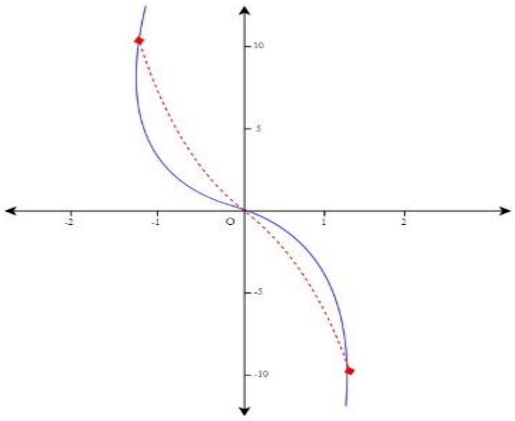

Fig 4 (b): Arc Tan activation

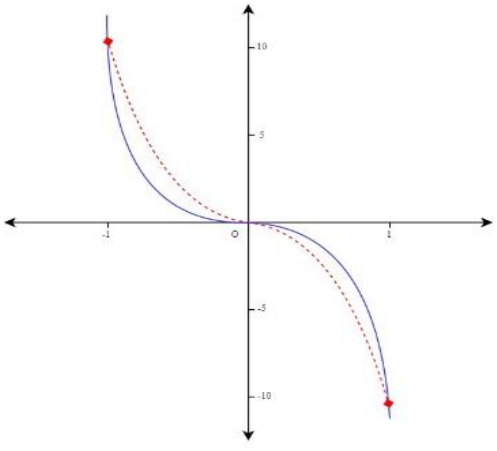

Fig 4 (c): Soft Sign activation

Where $\boldsymbol{m_1}$ represents the number of neurons in the hidden layer and $'\boldsymbol{M}'$ represents neuron in the output on replacing equation (4) with equation (3) we remain with the output in neuron $'\boldsymbol{k}$' defined with respect to equation (5) [22, 24, 27].

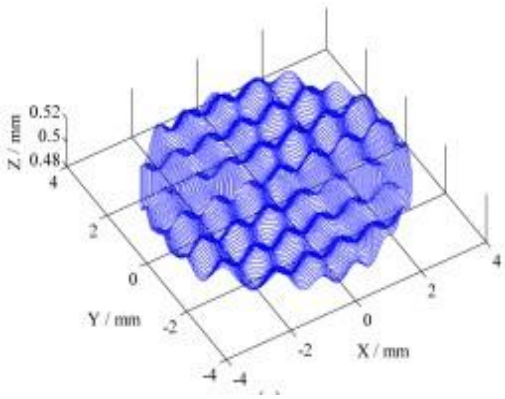

Fig 5 (a): Derivative of Arc Tan activation function

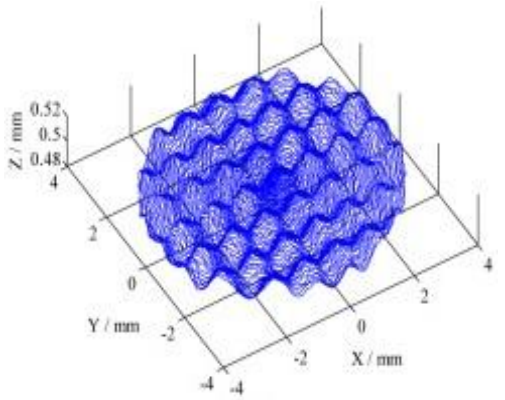

Fig 5 (b): Derivative of Sigmoid Activation function

$$y_1\mathrm{M(t)} = \sum_{j=1}^{m1} T\ f_1\left(\sum_{j=1}^{n} p + \Theta(\mathrm{k})\right) + \Theta(\mathrm{M}) \qquad (5)$$

$$\text{where } \mathrm{T} = w_k(\mathrm{M})$$

7. ANALYSIS OF THE PROPOSED IDEA

So, it would be possible to approximate the trained function with pre-trained multilayer perception with an identical activating function. An optimization problem with non-linear objective function is given; it's a cake-walk to solve the function in accordance with equation (5) as it turns out to be linear. Of course, we had approx. functions defined already; we couldn't stop it from turning into the hyperplane, so the activation function is nonlinear. Here, we selected Arc Tan function to easily simplify the function as its derivative is also simple [48, 49]. For example. Lagrange can solve trigonometric or exponential functions easily by turning it into polynomials. Fig 4(a) shows an Arc Tan activation function. It is similar to Fig 4(b) which is a sigmoidal function; as such, similar training is made on neuronal networks with both activation functions. The Sigmoidal function is used more often but here we use Arc Tan function because the derivative equation is simple compared to the expression of the sigmoidal. Though these are similar functions that are shown in Fig 5(a) and 5(b), while the derivate equation of Arc Tan is shown in the equation (6), and the derivative of sigmoidal can be obtained from equation (7) [46].

$$\arctan(f_1) = \frac{{f_1}'}{1 + {f_1}^2} \qquad (6)$$

$$\text{Sigmoid}(f_1) = -\frac{f_1' e^{f_1}}{(1+f_1)^2} \quad (7)$$

$f_1(y_1, y_2, \ldots\ldots\ldots, y_n)$ subject to

$$r_1(y_1, y_2, \ldots\ldots\ldots, y_n) \leq 0 \quad (8)$$

$\ldots\ldots\ldots\ldots$

$$\text{r}m_1(y_1, y_2, \ldots\ldots\ldots, y_n) \leq 0$$

Here, r_i is the constraint and f_1 is the objective function. Hence the approximation of DNN can be done using equation (5).

$$f_1(y_1, y_2, \ldots\ldots\ldots, y_n) = \sum_{j=1}^{m_1} T\ f_1\left(\sum_{j=1}^{n} p + \Theta(\text{k})\right) + \Theta(\text{M})$$

Subject to

$$r_1(y_1, y_2, \ldots\ldots\ldots, y_n) \leq 0 \quad (9)$$

$\ldots\ldots\ldots\ldots$

$$\text{r}m_1(y_1, y_2, \ldots\ldots\ldots, y_n) \leq 0$$

Hence, the subsequent optimization problems solve with Kuhn -Tucker, applying the equation.

$$\frac{\partial f_1(y_1, y_2, \ldots\ldots\ldots, y_n)}{\partial y_j} + \sum_{j=1}^{m_1} \lambda_j \frac{\partial r_j(y_1, y_2, \ldots\ldots\ldots, y_n)}{\partial y_j} \leq 0\ \forall\ k = 1, 2, 3, \ldots\ldots m_1$$

$$\lambda_1 * r_1(y_1, y_2, \ldots\ldots\ldots, y_n) = 0$$

$\ldots\ldots\ldots\ldots\ldots\ldots$

$$\lambda m_1 * \text{r}m_1(y_1, y_2, \ldots\ldots\ldots, y_n) = 0 \quad (10)$$

Defined the optimization problems with the Lagrange through equation (11)

$f_1(y_1, y_2, \ldots\ldots\ldots, y_n)$ subject to

$$r_1(y_1, y_2, \ldots\ldots\ldots, y_n) = 0 \quad (11)$$

$$\text{r}m_1(y_1, y_2, \ldots\ldots\ldots, y_n) = 0$$

According to optimization problems, it would be solved.

$$L_1(y_1, y_2, \ldots\ldots\ldots, y_n, \lambda_1, \lambda_2, \lambda_3, \ldots\ldots\ldots\ldots \lambda_{m_1}) = \sum_{j=1}^{m_1} T\ f_1\left(\sum_{j=1}^{n} p + \Theta(\text{k})\right) + \Theta(\text{M}) + \sum_{j=1}^{m_1} \lambda_i . r_i \frac{\partial L_1}{\partial y_j} = 0$$

$\forall\ j\ \in \{1, 2, 3\ldots\ldots\ldots n)$

$$\frac{\partial L_1}{\partial y_k} = 0\ \forall\ k \in \{1, 2, 3\ldots\ldots\ldots m_1) \quad (12)$$

From equation (10) and (12) we can say that it is essential to use the activation function to simplify the derivative and find the solution. Based on training made in the DNN, we can introduce a threshold for restrictions with inequality. For a lower threshold to be maintained, it is better to have a neural network that is prior trained with the variables around the defined constraints. In other terms for an obstacle with the same restrictions as equation (13) a set of data is generated to be used in the training process which is similar to the variable y_j. The change that is a difference in values that come one after the other and the error to define the threshold is in proportional.

$$a_j \leq y_j \leq b_j \quad (13)$$

$$y_j \geq a_j$$

$$y_j \leq b_j$$

Equation (15) defined here will be based on expression (5) and the restrictions defined according to equation (14) along with the threshold calculated.

$$\text{rs}(y_1, y_2, \ldots\ldots\ldots, y_n) \leq 0 \quad (14)$$

$$\sum_{j=1}^{m_1} T\ f_1\left(\sum_{j=1}^{n} p + \Theta(\text{k})\right) + \Theta(\text{M}) + \text{ß} \quad (15)$$

At last, expression (16) here is defined based on the optimization problem as in expression (8).

$$f_1(y_1, y_2, \ldots\ldots\ldots, y_n) = \sum_{j=1}^{B} A f_1\left(\sum_{j=1}^{n} C \cdot \text{D} + \text{E}\right) + F$$

Where:

$\text{A} = w_1\text{k}(\text{M}), \text{B} = m_1, \text{C} = w_1\text{j}(\text{k})$

$\text{D} = y_1\text{j}(\text{t}), \text{E} = \Theta(\text{k})$ and $\text{F} = \Theta(\text{M})$

Subject to:

$$\sum_{j=1}^{\boldsymbol{B}_1} A_1 f_1\left(\sum_{j=1}^{n_1} C_1 D_1 + E_1\right) + F_1 + \text{ß}_1 \quad (16)$$

$$\sum_{j=1}^{\boldsymbol{B}_2} A_2 f_1\left(\sum_{j=1}^{n_2} C_2 D_2 + E_2\right) + F_2 + \text{ß}_2$$

$\ldots\ldots\ldots\ldots\ldots$

$$\sum_{j=1}^{\boldsymbol{B}_m} A_m f_1\left(\sum_{j=1}^{n_m} C_m D_m + E_m\right) + F_m + \text{ß}_m$$

An Application of Support Vector Regression (SVR) is another technique to approximate functions which is an alternative to the use of neural networks. SVR approximates the functions as a linear combination in a higher dimension space than the original [25].

8. CONCLUSION

As to test the performance of recommendation, we evaluate different OP and compared the optimal values and predicted in the system. The neural network tool designed by our research group is used to test the optimization problems.

Under the domain of variables, the set of the dataset that we made use of training a neural network is generated objective function contains input variables and objective function obtained for these values contains output variables. The variable domain is defined on the constraints of the OP. Firstly, the system's performance was analyzed with a simple optimization problem. A multilayer perception is used to approximate a linear function by activating hidden and output linear layer functions.

minimum:

$$f(x) = ({x_1}^2 + {x_2}^2 - 1)^2$$

subject to:

$$-1 \le x_1 \le 1, -1 < x_2 \le 1 \qquad (17)$$

The objective function returned $\mathbf{1.34405}$ when $x_1 = -0.707$ and $x_2 = -0.707$ when it is replaced with the function that is approximated with the DNN. The result $\mathbf{1.34405}$ was of the objective function and the values of x_1 and x_1 is $\mathbf{-0.707}$. This was the result obtained by a quick training of neural networks; hence it could be improved easily.

Fig 6 (a) shows the original function whereas Fig 6 (b) shows an approximated one. It shows that the approximation worked well. There are two input layers in the neural network and seventeen neurons in the hidden layer, and Arc Tan activation function, and the objective function as an output. The momentum is $\mathbf{0.001}$ and the learning rate is $+\mathbf{0.01}$. The DNN is trained manually and is stopped when the constant is the error of $\mathbf{0.035}$, furthermore $\boldsymbol{y_1 = 0.46}$ and $\boldsymbol{y_1 = 0.3}$ were the results obtained from PSO. In this case, the results of the PSO algorithm worked well.

maximum:

$$\cos(x_1 \cdot x_2) = (x_1 + x_2)\frac{x_1 \cdot x_2}{100}\sin(x_1 + x_2)$$

subject to:

$$x_1 + x_2 \ge 0.5 \qquad (18)$$
$$x_1 \cdot x_2 \le 15$$
$$x_1 \ge 0$$
$$x_1 \le 1.5$$
$$x_2 \ge -1$$
$$x_2 \le 1$$

As we can see the result depicts multilayer perception with approximate objective functions and hence uses these results to solve OP. In a few instances when the new perspective calculated matches with the solution stated, meta-heuristics cannot provide a solution.

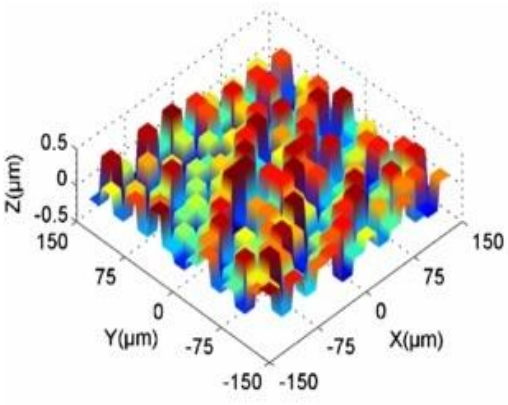

Fig 6(a): The original function

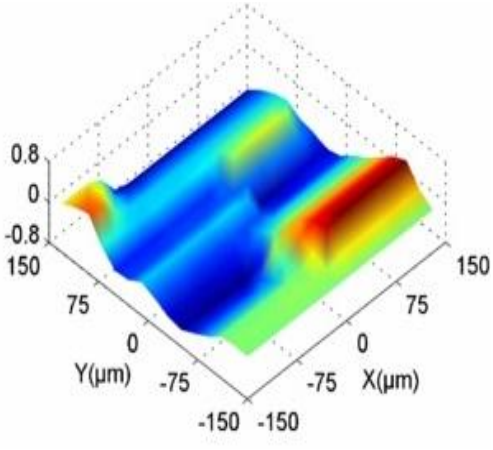

Fig 6 (b): The estimated function

The major drawback of this work is that it is necessary to instruct the DNN and therefore Lagrange or Kuhn- Tucker function is a mandatory usage with the DNN, this makes the function unfit for usage in any activation function like sigmoidal. Arctase function is used in this proposal as its derivative forms an equation that makes it relevant to solve the equation obtained by using Lagrange or Kuhn-Tucker. The major loophole of this work is that we need to come across a few errors when we deal with the approximation of equality constraints and hence the result stated will not be appropriate, while in other restrictions to obtain relevant solutions we introduce the lower limit and valid solution. Our experimental findings also demonstrate the overwhelming benefits of the resulting new attacks when attacking cryptographic systems that are both vulnerable and secure. We demonstrate that the neural networks can learn how to conduct forms of encryption and decryption, and also how to selectively apply these operations to achieve confidentiality goals.

9. ACKNOWLEDGMENTS

I would like to pay my sincere thanks and gratefulness to my academic advisor whose constant encouragement and support acted as an impetus for working hard and completing this paper with sincerity. Also, I would like to pay my heartiest gratitude to my family and to my alma mater, Department of Computer Science, Aligarh Muslim University for their unforgettable support whenever I needed. There are no words that can express gratitude for their love, affection and patience. They always stood by my side, had faith in my work and always prayed for my success.

10. REFERENCES

[1] Ríos-Mercado, Roger Z., and Conrado Borraz-Sánchez. "Optimization problems in natural gas transportation systems: A state-of-the-art review." Applied Energy 147 (2015): 536-555.

[2] Wang, Yong, et al. "Two-echelon logistics distribution region partitioning problem based on a hybrid particle swarm optimization–genetic algorithm." Expert Systems with Applications 42.12 (2015): 5019-5031.

[3] Salari, Majid, Mohammad Reihaneh, and Mohammad S. Sabbagh. "Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem." Computers & Industrial Engineering 83 (2015): 244-251.

[4] Zheng, Xiao-long, and Ling Wang. "A multi-agent optimization algorithm for resource constrained project scheduling problem." Expert Systems with Applications 42.15-16 (2015): 6039-6049.

[5] Lanza-Gutierrez, Jose M., and Juan A. Gomez-Pulido. "Assuming multiobjective metaheuristics to solve a three-objective optimisation problem for relay node deployment in wireless sensor networks." Applied Soft Computing 30 (2015): 675-687.

[6] Ploskas, Nikolaos, and Nikolaos Samaras. "Efficient GPU-based implementations of simplex type algorithms." Applied Mathematics and Computation 250 (2015): 552-570.

[7] Kuhn, H. W. "A. V7. Tucker,"Nonlinear Programming,"." Proc. Second Symp. on Mathematical Statistics and Probability, Univ. of Calif. 1951.

[8] Wang, Yong. "The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem." Computers & Industrial Engineering 70 (2014): 124-133.

[9] Li, Lin, Zhonghai Yu, and Yang Chen. "Evacuation dynamic and exit optimization of a supermarket based on particle swarm optimization." Physica A: Statistical Mechanics and its Applications 416 (2014): 157-172.

[10] Chen, Xiaohui, et al. "A ranging model based on BP neural network." Intelligent Automation & Soft Computing 22.2 (2016): 325-329.

[11] Choudhary, Priyankar, Vibhor Kant, and Pragya Dwivedi. "A Particle Swarm Optimization Approach to Multi Criteria Recommender System Utilizing Effective Similarity Measures." Proceedings of the 9th International Conference on Machine Learning and Computing. 2017.

[12] Jona, J., and N. Nagaveni. "A hybrid swarm optimization approach for feature set reduction in digital mammograms." WSEAS Trans. Inf. Sci. Appl 9.11 (2012): 340-349.

[13] Rohani, Abbas, Morteza Taki, and Masoumeh Abdollahpour. "A novel soft computing model (Gaussian process regression with K-fold cross validation) for daily and monthly solar radiation forecasting (Part: I)." Renewable Energy 115 (2018): 411-422.

[14] Zhang, Yuchen, and Lin Xiao. "Stochastic primal-dual coordinate method for regularized empirical risk minimization." The Journal of Machine Learning Research 18.1 (2017): 2939-2980.

[15] Choi, Jihun, Kang Min Yoo, and Sang-goo Lee. "Learning to compose task-specific tree structures." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[16] Chen, Pin-Yu, et al. "Ead: elastic-net attacks to deep neural networks via adversarial examples." Thirty-second AAAI conference on artificial intelligence. 2018.

[17] Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." arXiv preprint arXiv:1706.06083 (2017).

[18] Papernot, Nicolas, et al. "Distillation as a defense to adversarial perturbations against deep neural networks." 2016 IEEE Symposium on Security and Privacy (SP). IEEE, 2016.

[19] Papernot, Nicolas, et al. "Distillation as a defense to adversarial perturbations against deep neural networks." 2016 IEEE Symposium on Security and Privacy (SP). IEEE, 2016.

[20] Frank, Stephen, Ingrida Steponavice, and Steffen Rebennack. "Optimal power flow: a bibliographic survey II." Energy Systems 3.3 (2012): 259-289.

[21] Ma, Jiaqi, et al. "Parsimonious shooting heuristic for trajectory design of connected automated traffic part II: computational issues and optimization." Transportation Research Part B: Methodological 95 (2017): 421-441.

[22] Lin, Shen, and Brian W. Kernighan. "An effective heuristic algorithm for the traveling-salesman problem." Operations research 21.2 (1973): 498-516.

[23] Pouchet, Louis-Noël, et al. "Iterative optimization in the polyhedral model: Part II, multidimensional time." ACM SIGPLAN Notices 43.6 (2008): 90-100.

[24] Sun, Haoran, et al. "Learning to optimize: Training deep neural networks for wireless resource management." 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). IEEE, 2017.

[25] Snoek, Jasper, et al. "Scalable bayesian optimization using deep neural networks." International conference on machine learning. 2015.

[26] Snoek, Jasper, et al. "Scalable bayesian optimization using deep neural networks." International conference on machine learning. 2015.

[27] Lorenzo, Pablo Ribalta, et al. "Particle swarm optimization for hyper-parameter selection in deep neural networks." Proceedings of the genetic and evolutionary computation conference. 2017.

[28] Kraus, Mathias, Stefan Feuerriegel, and Asil Oztekin. "Deep learning in business analytics and operations research: Models, applications and managerial implications." arXivpreprint arXiv:1806.10897 (2018).

[29] Kraus, Mathias, Stefan Feuerriegel, and Asil Oztekin. "Deep learning in business analytics and operations research: Models, applications and managerial implications." arXivpreprint arXiv:1806.10897 (2018).

[30] Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." Proceedings of

the IEEE conference on computer vision and pattern recognition. 2015.

[31] Kudugunta, Sneha, and Emilio Ferrara. "Deep neural networks for bot detection." Information Sciences 467 (2018): 312-322.

[32] Yun, Chulhee, Suvrit Sra, and Ali Jadbabaie. "Global optimality conditions for deep neural networks." arXiv preprint arXiv:1707.02444 (2017).

[33] Li, Housen, et al. "NETT: Solving inverse problems with deep neural networks." Inverse Problems (2020).

[34] Samek, Wojciech, et al. "Evaluating the visualization of what a deep neural network has learned." IEEE transactions on neural networks and learning systems 28.11 (2016): 2660-2673.

[35] Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[36] Yosinski, Jason, et al. "How transferable are features in deep neural networks?." Advances in neural information processing systems. 2014.

[37] Adler, Jonas, and Ozan Öktem. "Solving ill-posed inverse problems using iterative deep neural networks." Inverse Problems 33.12 (2017): 124007.

[38] Maghrebi, Houssem, Thibault Portigliatti, and Emmanuel Prouff. "Breaking cryptographic implementations using deep learning techniques." International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Cham, 2016.

[39] Xie, Pengtao, et al. "Crypto-nets: Neural networks over encrypted data." arXiv preprint arXiv:1412.6181 (2019).

[40] Abadi, Martín, and David G. Andersen. "Learning to protect communications with adversarial neural cryptography." arXiv preprint arXiv:1610.06918 (2016).

[41] Berman, Daniel S., et al. "A survey of deep learning methods for cyber security." Information 10.4 (2019): 122.

[42] Sagduyu, Yalin E., Yi Shi, and Tugba Erpek. "IoT network security from the perspective of adversarial deep learning." 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). IEEE, 2019.

[43] Bae, Ho, et al. "Security and privacy issues in deep learning." arXiv preprint arXiv:1807.11655 (2018).

[44] Mariot, Luca, and Alberto Leporati. "Heuristic search by particle swarm optimization of boolean functions for cryptographic applications." Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation. 2015.

[45] Hassan, Shabbir, and Mohammad Ubaidullah Bokhari. "Computing in Cryptography." 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2016.

[46] Bokhari, M. U., and Shabbir Hassan. "A comparative study on lightweight cryptography." Cyber Security. Springer, Singapore, 2018. 69-79.

[47] Prof. M. U. Bokhari , Shabbir Hassan, 2020, Design of a Lightweight Stream Cipher: BOKHARI 256, International Journal of Engineering Research & Technology (IJERT) Volume 09, Issue 03 (March 2020).

[48] Shabbir Hassan, Prof. M. U. Bokhari, "Analysis and Design of LFSR Based Cryptographic Algorithm" in UGC Approved Journal Journal of Advances and Scholarly Researches in Allied Education (JASRAE) in Vol. 16, Issue No. 9, June-2019, ISSN 2230-7540 SCOPUS index..

[49] Shabbir Hassan, Prof. M. U. Bokhari, "Design of Pseudo Random Number Generator using Linear Feedback Shift Register" in UGC CARE Approved Journal of International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-2, December, 2019 in SCOPUS index.

11. AUTHORS PROFILE

Shabbir Hassan *Sun Certified Java Programmer (SCJP)* currently working as Assistant Professor at Centre for Distance Education, Aligarh Muslim University, Aligarh. He holds Master in Computer Science and Applications (MCA) and currently pursuing Ph.D. at Department of Computer Science, Aligarh Muslim University. His thrust area is "*Analysis and Design of Lightweight Stream Cipher*" and area of interest includes Applied Mathematics, Analysis and Design of Algorithms, Dynamic Programming, Network Security and Cryptography. He has qualified UGC-National Eligibility Test (NET) and has availed Junior Research Fellowship (JRF) during the research work. Throughout his career, he has been involved in innovative Software Development and Academic Teaching of Computer Science subjects like C, JAVA, Python, Data Structure, Operating System, Automata Theory and Computer Networks. He has presented his research work in several National and International IEEE Conferences and marked his active participation in many Conferences, Workshops and Symposia. His research papers have published in many reputed peer reviewed Journals of International repute like Springer, Elsevier, JASRAE, InderScience, UGC-CARE Journals and Scopus Indexed Database. Apart from the Academic Research and Software Development, he is enriched with the passion of poetry and philosophy and engages himself in social work