Hyperparameter optimization using custom genetic algorithm for classification of benign and malicious traffic on internet of things–23 dataset

Karthikayini Thavasimani, Nuggehalli Kasturirangan Srinath
Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering, Bengaluru, India

Article Info

Article history:

Received Jan 15, 2021
Revised Mar 21, 2022
Accepted Apr 5, 2022

Keywords:

Auto tuning hyperparameters
Bots
Custom genetic algorithm
Cyber-attacks
Hyperparameters
Neural networks
Optimization

ABSTRACT

Hyperparameter optimization is one of the main challenges in deep learning despite its successful exploration in many areas such as image classification, speech recognition, natural language processing, and fraud detections. Hyperparameters are critical as they control the learning rate of a model and should be tuned to improve performance. Tuning the hyperparameters manually with default values is a challenging and time-intensive task. Though the time and efforts spent on tuning the hyperparameters are decreasing, it is always a burden when it comes to a new dataset or solving a new task or improving the existing model. In our paper, we propose a custom genetic algorithm to auto-tune the hyperparameters of the deep learning sequential model to classify benign and malicious traffic from internet of things-23 dataset captured by Czech Technical University, Czech Republic. The dataset is a collection of 30.85 million records of malicious and benign traffic. The experimental results show a promising outcome of 98.9% accuracy.

Corresponding Author:

Karthikayini Thavasimani
Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering
Mysore Rd, RV Vidyaniketan, Post, Bengaluru, Karnataka 560059, India
Email: karthikayini@outlook.com

1. INTRODUCTION

A bot is a software program designed to automate tasks/run scripts/imitate human behavior such as content crawling, manipulating opinions, targeting and vulnerable machines. Initially, bots were created with good intentions to execute automated tasks such as search engine bots (Google/Bing), website monitoring bots (Pingdom), Social network bots (Facebook bot), and so on. These bots are naive, and it is easy to detect with standard detection strategies based on its executing nature. Now, bots are used in e-commerce, telemarketing, educational sectors for coaching, self-guided learning, and large enterprises as intelligent personal assistants like Google's Alexa, Apple's Siri, and social networks software development projects. They are beneficial to individuals as well as to businesses. As summarized in [1], bots are used for various purposes/tasks in GitHub projects. Despite the benefits, they are also used for exploiting information from the systems, threatening by hacking private information of the user, false allegations, fake advertisements, banking theft, launching distributed denial-of-service (DdoS) attacks, particularly on e-commerce sites, affecting the stock markets, and so on. Generally, Botnets, a network of compromised hosts or bots, are controlled by a bot-master to do these malicious activities. These bots can be detected using techniques such as a system based on social network information, crowdsourcing, or feature based. The network visualization of [2] explained the effects of bad bots that created online debate among Twitter users on recent California law on vaccination requirements.

Due to advancements encountered in information technology [3], increased usage of social networks, application programming interfaces (APIs), internet of things (IoT) smart devices, e-commerce, and cloud services, the number of cyber-attacks is increasing. Cyber-attacks target the vulnerable systems to either destroy the services or destroy valuable information or financial benefit. Various types of new cyberattacks are evolving every day on the internet. Cyberattacks are classified as software supply chain attacks, phishing attacks, cloud under attacks, and mobile devices [4]. For instance, attackers injected a skimming script into the JavaScript libraries used by online stores on PrismWEB, an e-commerce platform affecting more than 200 online university campus stores in North America. Personalizing the email contents or encoding emails with malicious links are social engineering methods used by the scammers to target personal accounts. This year 2020, has encountered many attacks on public cloud environments resulting in high data theft incidents across the business's world. Mobile device attacks, especially banking attacks, are the most contemporary cyber-attacks faced by individuals. Most of these attacks are triggered using bots.

As per the statistics, IoT devices are projected to reach 20 billion by 2021. Their usage in the information technology (IT) industries is expected to elevate in the coming years. There is a huge risk of bot attacks on these devices when the internet world is acclimating to the next stage of innovations. With the risks and the rising popularity coupled with broad adoption, versatility and diversity, and monotony restrictions, industries require new forms of protection to detect unauthorized internet of things devices in their network. Despite the research on existing technologies, it should take a wide-angle on recent technologies like IoT as there are more chances of a security breach. IoT devices include systems apart from computer systems such as smart speakers, smart lights, and smart lockers that exchange the data over the internet without human intervention. Due to minimal manual intervention, they are more vulnerable. It is easy to launch a DDoS attack from any controlled IoT device autonomously. After a while, it will behave like a regular device. The challenge lies in determining these devices that change their signature dynamically. Our work is organized into the following sequence, section 1 presents related work on the IoT device vulnerabilities and tuning a deep learning model’s hyper-parameter, section 2 is about the dataset and feature selection process, section 3 describes the proposed custom genetic algorithm (CGA) to auto-tune the hyper-parameters, section 4 discusses the experimental results and Section 5 includes conclusion and future work.

2. RELATED WORK

With rapid technological innovations and adoptions, the cybersecurity situation will be much worse considering the current deteriorating situation. The greatest security threat now may be due to internet of things devices [5]. The unprecedented risks that IoT devices pose to businesses are explained by many security experts. IoT devices are the ideal target of a botnet to launch DDoS attacks.

Mustapha and Alghamdi [6] explained the common vulnerabilities of IoT devices and provided ways to solve them. Insufficient authentication, authorization, modifying default passwords, monitoring ports, insecure network services, and lack of transport encryption. are some of the common reasons for security breaches in IoT devices [6]. The recent major technological trends in the industries are collaboration, automation, fighting the unknown, virtual and shared resources, and the internet of things progress towards an autonomous world [7]. So, finding and blocking the bots that control the IoT devices for malicious activities is vital. Recently, extensive research is carried out on how intelligent systems can address the problem of cybersecurity issues.

Machine learning and deep learning are the pathways to achieving cognitive intelligence in machines. Deep learning is a subset of machine learning, inspired by the artificial neural network, a collection of interconnected neurons trained to infer the results. Intelligent systems can detect security breaches in the IoT devices or any network devices per se [7]. Ranjit *et al.* [8] claimed that Bayesian optimization results in better hyperparameter space coverage with high validation accuracy in the cloud infrastructure. They used a long term convolutional recurrent neural network for tuning the hyperparameters in a distributed fashion and validated the results using video activity recognition problem. Neary [9] proposed a technique using reinforcement learning and convolutional neural network (CNN) to optimize the hyperparameters for object recognition. A multiagent-based method was used, and they reported an accuracy of 95% using the Modified National Institute of Standards and Technology dataset, where the optimal parameters were obtained in a short period.

Aich *et al.* [10] proposed a CNN-based model for extracting valuable information from various web content categories that the text mining applications can leverage. They manually tuned the hyperparameters of their deep learning model to improve the accuracy. By manually adjusting the hyperparameters, they achieved 85 to 92% accuracy in the text classification task. Finding an optimal hyperparameter, a time-consuming process, plays a crucial role in improving the model's accuracy. Ugli *et al.* [11] developed a system based on deep learning to detect and classify the source of the distractions while driving a vehicle in real-time to avoid accidents and to improve transport safety. A transfer learning method with pre-trained weights and ResNet50 using various optimizers is developed to enhance the model's learning capabilities. Stochastic gradient descent, Adam and RMSprop optimizers are selected after evaluating them manually to improve the accuracy. Their transfer learning system with SGD optimizer and ResNet50 model achieved an accuracy of 98.4% on the test dataset, "distracted drivers".

Chakraborty *et al.* [12] developed a 3D CNN architecture to learn the intricate patterns in the magnetic resonance imaging (MRI) scans to detect Parkinson's disease. The 3T T1-weighted data from MRI scans from the Parkinson's progression markers initiative database containing 406 subjects' information is used for the research purpose and the Bayesian optimization technique, Bayesian sequential model-based optimization (SMBO) is applied with a fixed penalty value, to tune the hyperparameters using the optimizer, "AdaDelta" to determine the optimal hyperparameters. The developed model achieved an overall accuracy of 95.29 with 0.943 as recall, 0.927 as precision and an F1 score of 0.936. Antonio [13] presented an optimization framework of a sequential model to optimize a multi-extremal, black box and a partially defined expensive objective function undefined outside the feasible regions. Subject vector machine constrained Bayesian optimization (SVM-BO) with two phases, one to approximate the unknown possible region boundary using subject vector machine and the other to find the globally optimal solution in the feasible region is applied. The Bayesian optimization process that uses a fixed penalty value is compared for analysis. The author concluded by showing the significant efficiency and computational efficiency over Bayesian optimization with SVM-BO.

Goel *et al.* [14] proposed OptCoNet, a CNN architecture for automatic screening of coronavirus disease (COVID-19) patients into three categories, normal, pneumonia and COVID-19 positive. The proposed architecture consists of optimized extraction of features and classification components. They used grey wolf optimizer (GWO) to find the optimal hyperparameters, publicly available X-ray images of normal, pneumonia and COVID-19 patients for analysis. They compared the results of the proposed architecture with existing optimized CNN methods. They concluded that OptCoNet outperformed other models with an accuracy of 97.78% and an F1 score of 95.25%. Abbas *et al.* [15] proposed a deep learning CNN based model called DeTraC that performs activities such as decomposition, transfer, and composition to classify the chest X-ray images of COVID-19 patients and deal with data irregularities. A comprehensive dataset of 80 samples of normal chest X-ray (CXR) images from the Japanese Society of Radiological Technology from various hospitals worldwide and 116 CXR images of COVID-19 and severe acute respiratory syndrome (SARS) was used for analysis. The hyperparameters of the models were manually selected and compared with the pre-trained models. An accuracy of 93.0% (with a sensitivity of 100%) in COVID-19 detection from X-ray images was reported.

Namasudra *et al.* [16] proposed a nonlinear autoregressive neural network time series (NAR-NNTS) model to the COVID-19 cases. Training algorithms such as Levenberg Marquardt for optimization with Bayesian regularization and scaled conjugate gradient was used. NAR-NNTS model outperforming others in forecasting the COVID-19 cases in two metrics, mean square error (MSE) and root mean square error (RMSE). Lin *et al.* [17] proposed a neural network model named attention segmental recurrent neural network (ASRNN) that uses a semi-Markov conditional random field (semi-CRF) model for sequence labelling. Using hierarchical structure and attention mechanism, the model differentiates more important and less important information while constructing the segments. Mini-batch stochastic gradient descent is used as the optimization algorithm with limited static hyperparameters. In CoNLL2000 and Cora dataset, the proposed model outperformed existing methods with an F1 score of 93.7 and 80.18%.

Yi and Bui [18] proposed a framework fusing meta-learning and Bayesian technique to predict better highway traffic. They used the Korean highway system dataset for analysis by optimizing the hyperparameters. Bui and Yi [19] optimized hyperparameters of deep learning models using meta-learning to predict the traffic with data from the vehicle detection system. Hoof *et al.* [20] proposed a new surrogate model based on gradient boosting to find the optimal hyperparameters and regulated the exploration space with success on the reasonably sized set's classification problems. However, the proposed model is not built and scaled for the real work problems with larger datasets such as network data. Multi-objective simulated annealing (MOSA) [21] algorithm that efficiently searches the objective space outperforming the simulated annealing (SA) algorithm with a caveat that the computational complexity is as important as the test accuracy. Hoopes *et al.* [22] proposed HyperMorph, a learning-based strategy that eliminates the need to tune hyperparameters during training, reducing the computational time but limits the capability to find the optimal values.

Lee *et al.* [23] applied the genetic algorithm on CNN using both network configuration and hyperparameters. They showed that their algorithm outperformed genetic CNN by 11.74% on an amyloid brain image dataset used for Alzheimer's disease diagnosis. Lin *et al.* [24] proposed a hybrid approach by combining a modified hyperband, mode-pursuing sampling in the trust-region, and repeating the selection and sampling process until a termination criterion is met for hyperparameter optimization. Shaziya *et al.* [25]

explored the impacts of the various hyperparameters in developing the deep learning model. The proposed system aims to classify the intrusion attacks on the cyber defense dataset (CSE-CIC-IDS2018) by applying artificial intelligence and achieved an accuracy result as 99.9% and receiver operating characteristic as 0.999% respectively [26]. Hossain *et al.* [27] achieved 99.08% accuracy, and a detection rate of 0.93 on Intrusion Detection Evaluation Dataset 2017, a labelled dataset to detect DoS attacks by fine-tuning long short term memory hyperparameters: learning rates, loss functions, optimizers, and activation functions.

To reduce DenseNet convolutional networks' training time, Nugroho and Suhartanto [28] have used a random search to select the best candidates for batch size and learning rate. Thavasimani and Srinath [29] analyzed various optimizers and their impact on a deep learning model's performance and concluded the importance of tuning the hyperparameters. Kimura *et al.* [30] tuned the CASIA algorithm's hyperparameters to increase the detection of classification rate, effectively improving the iris liveness detection. Various hyperparameters of a random search algorithm were analyzed to achieve the best performance [31]. A reinforcement learning mechanism is applied to tune the hyperparameters to control computational resource allocation problems [32]. Complete insights into the recent advancements and theories underlying Bayesian optimization and explains how it solves the difficult problems were discussed.

Kiatkarun and Phunchongharn [33] proposed hyper-genetic, a variant for the genetic algorithm to obtain the optimal hyperparameters for gradient boosting machine algorithm. They further compared their results with grid search, Bayesian optimization, grid search and random searching techniques. They showed their results were outperforming the others when they evaluated on four different datasets. Likewise, Putatunda and Rama [34] proposed and proved that the randomized-hyperopt outperforms hyperopt, grid and random search techniques. Huang *et al.* [35] solved the capacitated arc routing problem in logistics using Bayesian optimization technique. Jamieson *et al.* [36] applied Bayesian optimization on kernel ridge regression machine learning methods to find optimal hyperparameter configuration.

Cho *et al.* [37] proposed a modified version of Bayesian optimization: diversified, early termination enabled, and parallel Bayesian optimization (DEEP-BO), to select the optimal hyperparameters and benchmarked the performance with existing techniques. DEEP-BO mostly outperformed other methods but was not optimal in choosing the best hyperparameters. Bayesian optimization dominates the optimization techniques in the current scenario. It focuses to optimize the config selection to solve the problem of optimizing the function that are high dimension and non-convex with possibly less evaluation noise and unknown smoothness. The above papers and current black box techniques motivated us to create a custom genetic algorithm (CGA), an optimization technique to return the best hyperparameters of deep learning models with higher prediction accuracy.

3. DATA PREPROCESSING

For our investigation, we have used the IoT-23 dataset [38] containing malicious and benign traffic. It is a new dataset captured on IoT devices with 20 captures of malware attacks and three captures of benign traffic accounting for 21 GB in size. It was first published in the year, January 2020. The dataset contains data monitored from 2018 through 2019 in the Stratosphere Laboratory, Artificial Intelligence Center group, Faculty of Electrical Engineering, Czech Technical University, the Czech Republic, and funded by Avast and Prague. The main goal of capturing this IoT dataset is to provide a platform for the researchers to efficiently test their algorithms/models. The IoT devices used in the laboratory are an Amazon Echo home intelligent personal assistant, Philips HUE smart LED lamp, and Somfy smart door lock. The official site, stratosphere.org contains comprehensive information on the types of malicious attacks and benign traffic listed in Table 1 and Table 2, and how they were collected.

Table 1. Summary of the benign scenarios

#	Name of Dataset	Duration(~hrs)	Packets	ZeekFlows	Pcap Size	Device
1	CTU-Honeypot-Capture-7-1 (somfy-01)	1.4	8,276	139	2,094 KB	Somfy Door Lock
2	CTU-Honeypot-Capture-4-1	24	21,000	461	4,594 KB	Philips HUE
3	CTU-Honeypot-Capture-5-1	5.4	398,000	1,383	381 MB	Amazon Echo

The dataset contains malicious traffic captured at different protocols such as domain name system (DNS), hypertext transfer protocol (HTTP), dynamic host configuration protocol (DHCP), telecommunication network (Telnet), and secure socket layer (SSL), We used the labeled connection log file that fields such as *uid*, *id.orig_h*, *id.orig_p*, *id.resp_h*, *id.resp_p*, *proto*, *service*, *duration*, *orig_bytes*, *resp_bytes*, *conn_state*, *local_orig*, *local_resp*, *missed_bytes*, *history*, *orig_pkts*, *orig_ip_bytes*, *resp_pkts*, *resp_ip_bytes, tunnel_parents* and *target output*. All the files are consolidated into a single file for preprocessing.

The file contents are split into thousand rows using threads and the missing values are filled. Empty, null, and hyphen values are replaced with zero and non-string values with not available (NA). The strings values are converted to numerical values using the one-hot encoding technique and label encoder (categorical values).

Table 2. Summary of malicious IoT scenarios

#	Name of Dataset	Duration (hrs)	Packets	ZeekFlows	Pcap Size	Name
1	CTU-IoT-Malware-Capture-34-1	24	233,000	23,146	121 MB	Mirai
2	CTU-IoT-Malware-Capture-43-1	1	82,000,000	67,321,810	6 GB	Mirai
3	CTU-IoT-Malware-Capture-44-1	2	1,309,000	238	1.7 GB	Mirai
4	CTU-IoT-Malware-Capture-49-1	8	18,000,000	5,410,562	1.3 GB	Mirai
5	CTU-IoT-Malware-Capture-52-1	24	64,000,000	19,781,379	4.6 GB	Mirai
6	CTU-IoT-Malware-Capture-20-1	24	50,000	3,210	3.9 MB	Torii
7	CTU-IoT-Malware-Capture-21-1	24	50,000	3,287	3.9 MB	Torii
8	CTU-IoT-Malware-Capture-42-1	8	24,000	4,427	2.8 MB	Trojan
9	CTU-IoT-Malware-Capture-60-1	24	271,000,000	3,581,029	21 GB	Gagfyt
10	CTU-IoT-Malware-Capture-17-1	24	109,000,000	54,659,864	7.8 GB	Kenjiro
11	CTU-IoT-Malware-Capture-36-1	24	13,000,000	13,645,107	992 MB	Okiru
12	CTU-IoT-Malware-Capture-33-1	24	54,000,000	54,454,592	3.9 GB	Kenjiro
13	CTU-IoT-Malware-Capture-8-1	24	23,000	10,404	2.1 MB	Hakai
14	CTU-IoT-Malware-Capture-35-1	24	46,000,000	10,447,796	3.6G	Mirai
15	CTU-IoT-Malware-Capture-48-1	24	13,000,000	3,394,347	1.2G	Mirai
16	CTU-IoT-Malware-Capture-39-1	7	73,000,000	73,568,982	5.3GB	IRCBot
17	CTU-IoT-Malware-Capture-7-1	24	11,000,000	11,454,723	897 MB	Linux.Mirai
18	CTU-IoT-Malware-Capture-9-1	24	6,437,000	6,378,294	472 MB	Linux.Hajime
19	CTU-IoT-Malware-Capture-3-1	36	496,000	156,104	56 MB	Muhstik
20	CTU-IoT-Malware-Capture-1-1	112	1,686,000	1,008,749	140 MB	Hide and Seek

We applied Chi-squared technique to determine the important features. Features such as *id.orig_h, id.orig_p, id.resp_h, id.resp_p, proto, service, duration, orig_bytes, resp_bytes, conn_state, history, orig_pkts, orig_ip_bytes, resp_pkts, resp_ip_bytes, and tunnel_parents* are marked as important for prediction. We added additional column, target output containing the label such as benign or malicious. Features such as *ts, uid, local_orig, local_resp* and *missed_bytes* are ignored. The final input file is used for training and testing the proposed custom genetic algorithm.

4. PROPOSED WORK

We propose a CGA for tuning the deep learning hyperparameters. Our algorithm is inspired by Darwin's theory of natural evolution. Evolutionary algorithms are widely used in search and optimization problems. So, we created CGA to address the limitations of the genetic algorithm. CGA aims to converge towards global optimum rather than local optima and problems with fitness measures resulting in poor convergence. We introduce three modules with simple approaches, bonus-penalty, retain and optimal selection, to increase the mutation rate and convergence towards the global optimum.

In the Bonus-penalty module, the algorithm applies penalties for networks reaching early convergence. Otherwise, the bonus is applied. The networks with bonus vales are selected to generate future offspring (child neural networks). Messenger ribonucleic acid (mRNA) is a single-stranded molecule of ribonucleic acid. In the gene, it corresponds to the genetic sequences. In the process of protein synthesis, they are read by ribosomes. In CGA, the bonus and penalty are routed as mRNA values and are read by the neural networks while creating new populations. Based on the bonus and penalty values, the population is either selected or discarded. In the retain module, the algorithm picks a random set of individuals from the discarded list post crossover and mutation to increase the mutation rate.

As a result, a wide range of hyperparameter values is explored in the search space. In optimal selection module, the top 50% of the high performing configurations in terms of less loss, high accuracy and high F1 measure are used for creating future off springs. This results in reducing the optimization time to a greater extent.

5. ALGORITHM DESCRIPTION

A deep learning sequential model with hyperparameter values from the Table 3 is used to initiate the process. The mRNA values, bonus and penalty are manually set to null in the beginning. As the population grows, these values change dynamically. The initial generation of neural network (NN) starts with configuration/hyperparameters space created using the module 3. A combination of metrics such as high F1 measure, high accuracy and low loss is used as a fitness function to select the best neural networks for creating child neural networks. NNs with higher metric values and bonus are preferred during the parent's selection randomly. So, there is a probability of using identical NNs. We eliminate this scenario by choosing random NNs. New child neural networks are created using crossover and mutation. If they converge early, the penalty is applied to the NNs. Otherwise, a bonus is applied. This process continues for any number of set generations. In our analysis, we used 20 generations. As a result of evolution, we obtain the optimal hyperparameter values with the highest metrics values.

5.1. Module 1 (M1) retain

To produce fittest off springs of neural networks (NNs), the parents are chosen with high accuracy, f1 measure and low loss and the remaining networks are eliminated. This results in early convergence at faster pace. To stop the premature convergence, mutation and diversity should be improved. To do so, a part of discarded neural networks is added back to the parent population.

5.2. Module 2 (M2) penalty-bonus

To maximize the convergence towards the problem’s global optimum, bonuses and penalties are used. The NNs that reach the convergence early are penalized and if they do not, bonuses are applied. While generating the future off springs, NNs with bonus are preferred. Using mRNA, the bonus and penalty values are routed/applied. Thus, mRNA values of bonus are prioritized in population selection.

5.3. Module 3 (M3) optimal selection

The initial and future populations of NNs are determined using optimal selection. The process starts with default population. The random parameters selected for initial population will significantly impact the overall process of finding the optimal hyperparameters. It might be slower, moderate, or faster in finding the optimal hyperparameters. One of the biggest challenges of genetic algorithms are the execution that depends on the populations to traverse, space to search and available resources.

Module 3 - Population selection
```
Input: Limit K, Set s where $j_{r,q}$ denotes the qth loss
From the rth set, Max size M
Initialize:
$S_0$ = [n], N=10
Finding the optimal set:
For j $\leftarrow$ 0,1…. c
Set $S_j$ = [$sN^{-j}$], $m_j$ = [$MN^{j-c}$]
Pull each set in $S_j$ for $m_j$ times
Keep the best set in terms of $m_j$th
observed loss as $S_{j+1}$
Output: Optimal set, $O_s$
```

The module 3 consists of steps, i) select a random hyperparameter configuration, ii) evaluate the performance, iii) use the top 50% and the rest are ignored, and iv) jump to step 2 until the optimal population is created (default: 10). In module 3, only 50% of the population is used. It is based on the assumption that promising population tend to produce better offspring than the worst ones even early in the process. Considering all the neural networks increases the execution time and the time taken to find the optimal hyperparameters. The objective is to avoid wasting the training time that will lead us nowhere and focus on the top 50% that is likely to yield optimal hyperparameters, thereby allocating resources to a potentially more valuable population. In CGA, the initial 50% of top-performing neural network hyperparameters are used to explore the hyperparameter space further, and the remaining NNs are ignored.

The focus is on reduction of training time of the hyperparameters that does not provide concrete solution. However, the top performing hyperparameter sets are used for further analysis. The module 3 is different from other approaches by i) limiting the configs to converge to reduce the usage of resources and ii) faster execution time by using top performing sets. Figure 1 contains the algorithm of the custom genetic algorithms with module 1, module 2 and module 3 and how they are invoked.

Figure 2 shows the workflow of the custom genetic algorithms. Start the process with initialization by setting the initial hyperparameters and the algorithm variables to null. Create the ancestors by invoking module 3, optimal selection, which returns the default population to start the evolution. They become the first generation of population/hyperparameters.

Custom Genetic Algorithm

```
Initialization
    Set the hyperparameters such as epochs, layers, and neurons,
    Set the penalty and bonus to 0
    Set earlystop with patience to 5
Create the initial population
    For I ←1 to size_of_the_population
        Create the network
        End for
        **Invoke Module 3 to return the optimal
        Neural Network to begin with**
    Return the network object (population)

For I ←1 to number_of_generations
    Train the networks
    Evolve the population of networks
        Score and sort the networks in descending order
        Select the parents with high score & priority for crossover
        While len(children) < desired_length:
            If same networks:
                Get a random parent network from population
                Breed and mutate
                Child network is formed
                Don't grow the children than desired length
        **#Module 2**
        If the network results in early stop:
            Add penalty
        If the network doesn't result in early stop:
            Add bonus
    Update the network's priority list with penalty and bonus
    Push the unused networks to discarded list

    **Invoke Module 3 to generate the optimal population set**

    **#Module 1**
    For I ←1 to retain_length:
        #Keep some random individuals for future crossover
        If random_select > random.random():
            population.append(individual)

    Select the hyperparameters with highest value
    (High Accuracy, Low loss and High f1 measure)
```

Figure 1. Custom genetic algorithm

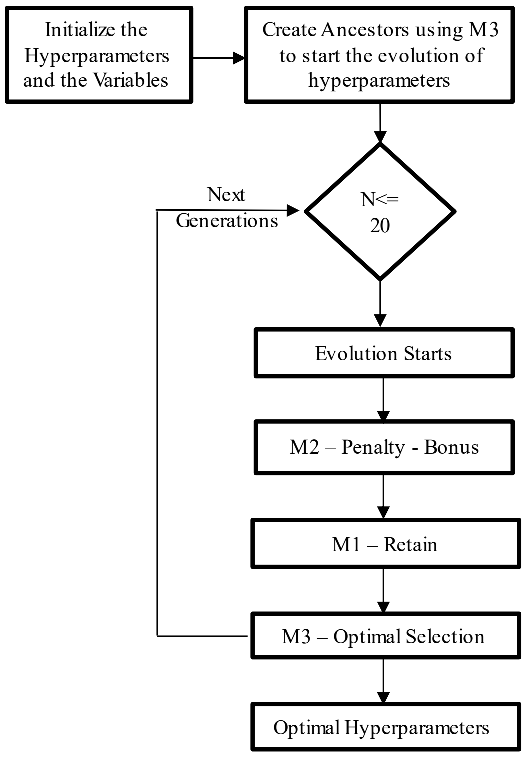
Initialize the Hyperparameters and the Variables
Create Ancestors using M3 to start the evolution of hyperparameters
Next Generations
N<= 20
Evolution Starts
M2 – Penalty - Bonus
M1 – Retain
M3 – Optimal Selection
Optimal Hyperparameters

Figure 2. Workflow of custom genetic algorithm

Once the evolution starts, prefer the parents with higher accuracy for crossover and mutation. Select a part of discarded individuals for diversity and mutation. Apply penalties to parents resulting in early convergence and bonuses to the parents with higher accuracy and diversity to select the optimal population. Repeat the process for twenty generations to obtain the optimal child. The final output is the model's optimal hyperparameters of the neural network.

6. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we present the experimental results. We tested our algorithm using Google Colab Pro with 25 GB RAM, 147 GB hard disk drive, and NVIDIA TESLA P100 GPU. Table 3 lists the hyperparameters, their range, and optimized values of default and CGA model. After iterating through 20 generations (with 50 populations per generation) on the hyperparameter space, the CGA returned the optimal hyperparameter values with 98.9% accuracy. For comparison, we experimented on a deep learning model with default hyperparameter values and other optimization techniques. The default hyperparameters resulted in an accuracy of 88.22%. The results of the default model will vary depending on the parameters. We validated the results on a default input range for our testing purpose using the brute force technique to avoid overfitting and underfitting.

Table 4 compares the default and CGA tuned models on various metrics such as accuracy, loss, precision, recall, and F1 score on training, validation, and test datasets. We used cross-validation split, 0.3 to ensure that the deep learning models generalize well to produce the best predictive model. We used 70% data for training and 30% for testing. CGA tuned model outperformed the default model with an accuracy of 98.9% on testing data.

Table 3. List of hyperparameters initial range settings and optimized values

Hyperparameters	Input range	Optimized values	
		Without tuning	CGA
Number of epochs	[20, 40, 50, 60]	40	20
Batch size	[20, 50, 100, 200, 300, 400]	500	20
Number of layers	[1, 2, 5, 10]	2	4
Number of neurons	[2, 5, 10, 15]	40	40
Regularizer	Dropout [0.1, 0.2, 0.5]	0.2	0.2
Optimizers	["adam", "sgd", "rmsprop", "Adadelta", "Adamax", "Nadam"]	Adam	Adam
Activations	["Relu", "sigmoid", "tanh"]	Relu	Relu
Last layer activation function	Sigmoid	Sigmoid	Sigmoid
Losses	Binary cross entropy	Binary cross entropy	Binary cross entropy
Metrics	Test accuracy	0.8822	0.9890

Table 4. Model results

	Training results					Validation results				
	Accuracy	Loss	Precision	Recall	F1	Accuracy	Loss	Precision	Recall	F1
Default	0.8851	0.2333	0.8274	0.9922	0.9022	0.8836	0.2251	0.8515	0.9483	0.8971
CGA	0.9675	0.1142	0.9523	0.9939	0.9703	0.9888	0.0535	0.9805	0.9991	0.9892

	Test Results				
	Accuracy	Loss	Precision	Recall	F1
Default	0.8822	0.2262	0.8489	0.9487	0.8932
CGA	**0.9890**	**0.0522**	0.9807	0.9991	**0.9895**

Figure 3(a)-(d) shows the accuracy and loss graphs of default and CGA tuned model. From the graph, it is evident that the CGA tuned model's accuracy increases as the loss decreases. The CGA tuned models eliminate overfitting, as well as underfitting, resulted in higher accuracy and lower loss. There is an increase of more than 10% in both training and test accuracy than models with default hyperparameter values. Figures 4(a)-(d) represents the receiver operating characteristic (ROC) curve of optimization techniques such as random, Bayesian, genetic algorithm, and custom genetic algorithm. The area under the curve for CGA is 0.99 that outperforms the existing optimization others in identifying the false negatives and true positives. Table 5 compares various metrics such as F-measure, accuracy, recall and precision for the existing optimization techniques such as random search, genetic algorithm, and Bayesian search. CGA tuned model outperformed the existing techniques and methods with an accuracy of 0.9890 and F-measure of 0.9895.

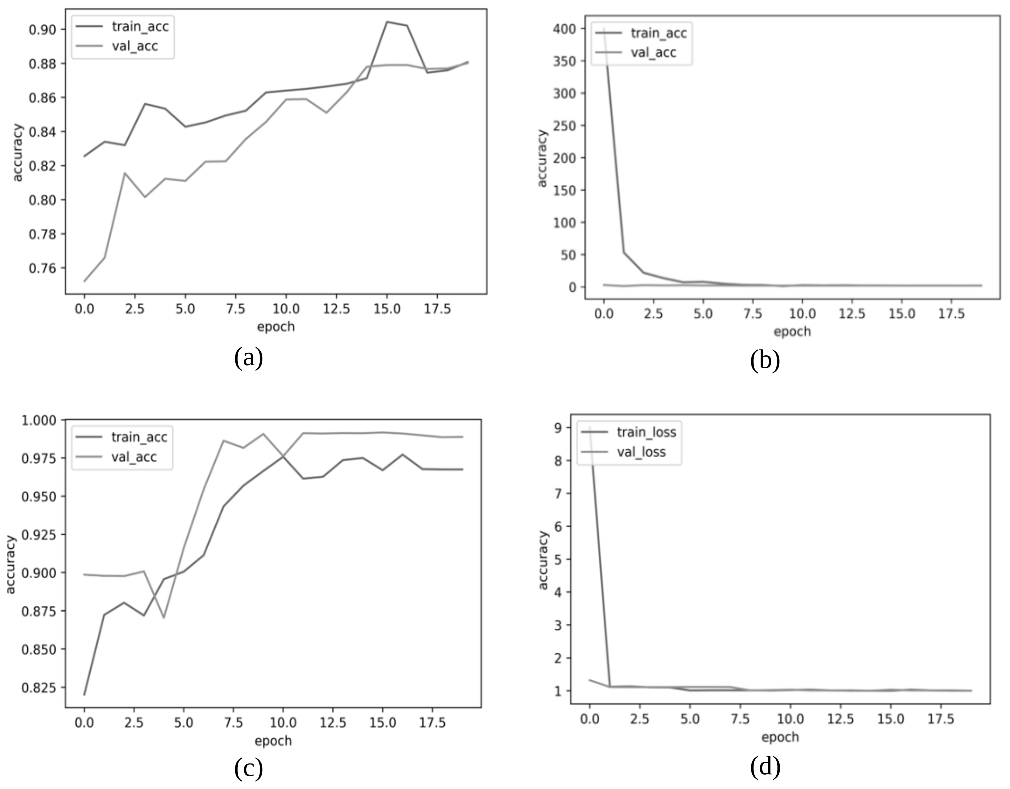

Figure 3. Comparing Default model's (a) accuracy, (b) loss with CGA tuned models, (c) accuracy and (d) loss

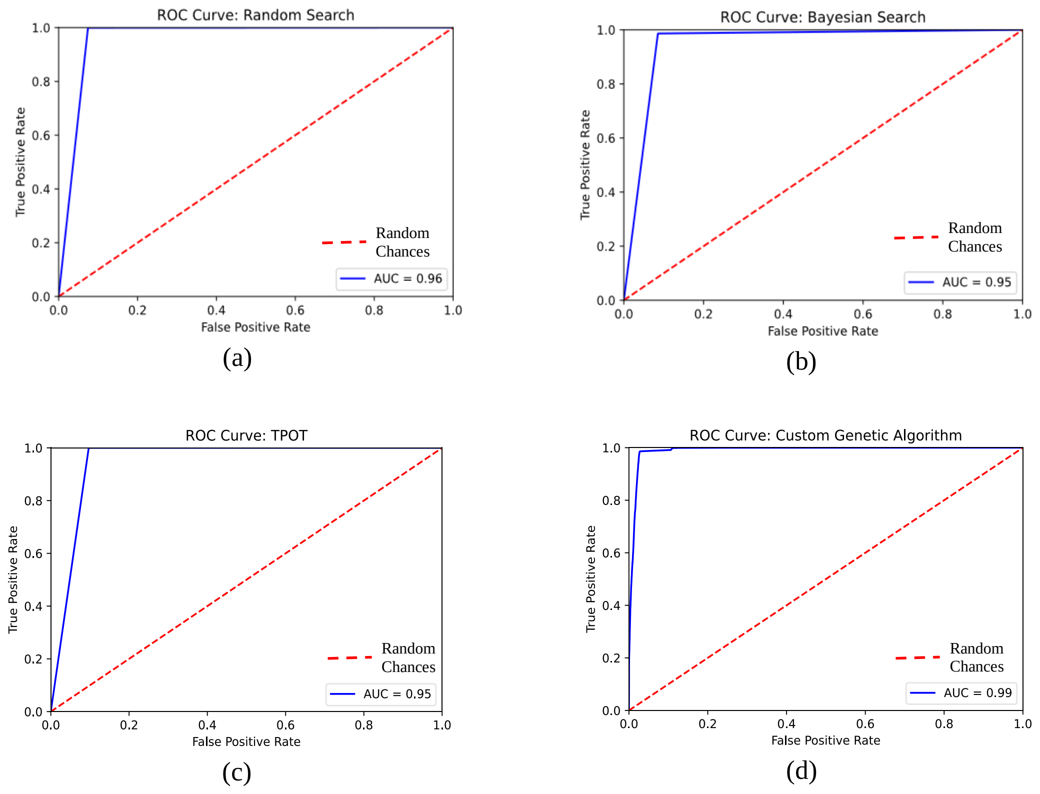

Figure 4. ROC curve for (a) random search, (b) Bayesian search, (c) genetic algorithm and (d) custom genetic algorithm

Table 5. Comparison with existing optimization techniques

Technique	Accuracy	F measure	Precision	Recall
Random Search	0.9650	0.9694	0.9411	0.9994
Bayesian Search	0.9535	0.9585	0.9322	0.9863
TPOT	0.9557	0.9609	0.9247	1.0000
CGA	0.9890	0.9895	0.9807	0.9991

7. CONCLUSION

In this paper, we discussed IoT devices, their vulnerabilities, the impact of malicious bots, and explained how a deep learning model is tuned to classify benign and malicious traffic. One of the biggest challenges of deep learning model is finding the optimal hyperparameter values. Thus, we proposed a custom genetic algorithm, an optimization technique to obtain the best hyperparameters for a model. The algorithm was able to successfully learn with a wide range of hyperparameter values without manual intervention. The CGA tuned model outperformed the default model with an accuracy of 98.9% and effective in reaching the optimal solution. As future work, we are planning to optimize the algorithm further by experimenting on new datasets and comparing it against other optimization techniques.

REFERENCES

[1] M. Wessel *et al.*, "The power of bots," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, pp. 1–19, Nov. 2018, doi: 10.1145/3274451.

[2] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, Jun. 2016, doi: 10.1145/2818717.

[3] D. K. Saini, "Sense the future," *Campus*, vol. 1, no. 11, pp. 14–17, 2011.

[4] S. Rokka Chhetri and M. A. Al Faruque, "Data-driven kinetic cyber-attack detection," Cham: Springer International Publishing, 2020, pp. 91–109.

[5] T. J. OConnor, W. Enck, and B. Reaves, "Blinded and confused," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, May 2019, pp. 140–150, doi: 10.1145/3317549.3319724.

[6] H. Mustapha and A. M. Alghamdi, "DDoS attacks on the internet of things and their prevention methods," in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, Jun. 2018, pp. 1–5, doi: 10.1145/3231053.3231057.

[7] Y. Harel, I. Ben Gal, and Y. Elovici, "Cyber security and the role of intelligent systems in addressing its challenges," *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 4, pp. 1–12, Jul. 2017, doi: 10.1145/3057729.

[8] M. P. Ranjit, G. Ganapathy, K. Sridhar, and V. Arumugham, "Efficient deep learning hyperparameter tuning using cloud infrastructure: intelligent distributed hyperparameter tuning with bayesian optimization in the cloud," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, Jul. 2019, vol. 2019-July, pp. 520–522, doi: 10.1109/CLOUD.2019.00097.

[9] P. Neary, "Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning," in *2018 IEEE International Conference on Cognitive Computing (ICCC)*, Jul. 2018, pp. 73–77, doi: 10.1109/ICCC.2018.00017.

[10] S. Aich, S. Chakraborty, and H.-C. Kim, "Convolutional neural network-based model for web-based text classification," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 5191–5785, Dec. 2019, doi: 10.11591/ijece.v9i6.pp5785-5191.

[11] I. K. K. Ugli, S. Aich, H. Ryu, M.-I. Joo, and H.-C. Kim, "Detection of distracted driving using deep learning," in *International Conference on Future Information and Communication Engineering*, vol. 12, no. 1, pp. 29–32.

[12] S. Chakraborty, S. Aich, and H.-C. Kim, "Detection of Parkinson's disease from 3T T1 weighted MRI scans using 3D convolutional neural network," *Diagnostics*, vol. 10, no. 6, Jun. 2020, doi: 10.3390/diagnostics10060402.

[13] C. Antonio, "Sequential model based optimization of partially defined functions under unknown constraints," *Journal of Global Optimization*, vol. 79, no. 2, pp. 281–303, Feb. 2021, doi: 10.1007/s10898-019-00860-4.

[14] T. Goel, R. Murugan, S. Mirjalili, and D. K. Chakrabartty, "OptCoNet: an optimized convolutional neural network for an automatic diagnosis of COVID-19," *Applied Intelligence*, vol. 51, no. 3, pp. 1351–1366, Mar. 2021, doi: 10.1007/s10489-020-01904-z.

[15] A. Abbas, M. M. Abdelsamea, and M. M. Gaber, "Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network," *Applied Intelligence*, vol. 51, no. 2, pp. 854–864, Feb. 2021, doi: 10.1007/s10489-020-01829-7.

[16] S. Namasudra, S. Dhamodharavadhani, and R. Rathipriya, "Nonlinear neural network based forecasting model for predicting COVID-19 cases," *Neural Processing Letters*, Apr. 2021, doi: 10.1007/s11063-021-10495-w.

[17] J. C.-W. Lin, Y. Shao, Y. Djenouri, and U. Yun, "ASRNN: a recurrent neural network with an attention model for sequence labeling," *Knowledge-Based Systems*, vol. 212, Jan. 2021, doi: 10.1016/j.knosys.2020.106548.

[18] H. Yi and K.-H. N. Bui, "An automated hyperparameter search-based deep learning model for highway traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5486–5495, Sep. 2021, doi: 10.1109/TITS.2020.2987614.

[19] K.-H. N. Bui and H. Yi, "Optimal hyperparameter tuning using meta-learning for big traffic datasets," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Feb. 2020, pp. 48–54, doi: 10.1109/BigComp48618.2020.0-100.

[20] J. van Hoof and J. Vanschoren, "Hyperboost: hyperparameter optimization by gradient boosting surrogate models," *arxiv.org/abs/2101.02289*, 2021.

[21] A. Gülcü and Z. Kuş, "Multi-objective simulated annealing for hyper-parameter optimization in convolutional neural networks," *PeerJ Computer Science*, vol. 7, Jan. 2021, doi: 10.7717/peerj-cs.338.

[22] A. Hoopes, M. Hoffmann, B. Fischl, J. Guttag, and A. V Dalca, "HyperMorph: amortized hyperparameter learning for image registration," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12729, Springer International Publishing, 2021, pp. 3–17.

[23] S. Lee, J. Kim, H. Kang, D.-Y. Kang, and J. Park, "Genetic algorithm based deep learning neural network structure and hyperparameter optimization," *Applied Sciences*, vol. 11, no. 2, Jan. 2021, doi: 10.3390/app11020744.

[24] J. Lin, H. Li, Y. Huang, J. Chen, P. Huang, and Z. Huang, "An efficient modified hyperband and trust-region-based mode-pursuing sampling hybrid method for hyperparameter optimization," *Engineering Optimization*, vol. 54, no. 2, pp. 252–268, Feb. 2022, doi: 10.1080/0305215X.2020.1862823.

[25] H. Shaziya and R. Zaheer, "Impact of hyperparameters on model development in deep learning," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 56, Springer Singapore, 2021, pp. 57–67.

[26] V. Kanimozhi and T. P. Jacob, "Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," in *International Conference on Communication and Signal Processing (ICCSP)*, Apr. 2019, pp. 33–36, doi: 10.1109/ICCSP.2019.8698029.

[27] M. D. Hossain, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based network attack detection: performance comparison by hyper-parameter values tuning," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, Aug. 2020, pp. 62–69, doi: 10.1109/CSCloud-EdgeCom49738.2020.00020.

[28] A. Nugroho and H. Suhartanto, "Hyper-parameter tuning based on random search for DenseNet optimization," in *2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Sep. 2020, pp. 96–99, doi: 10.1109/ICITACEE50144.2020.9239164.

[29] K. Thavasimani and N. K. Srinath, "Deep learning techniques: a case study on comparative analysis of various optimizers to detect bots from CRESCI-2017 dataset," *International Journal of Advanced Science and Technology*, vol. 29, no. 4, pp. 10040–10053, 2020.

[30] G. Kimura, D. Lucio, A. Britto Jr., and D. Menotti, "CNN hyperparameter tuning applied to iris liveness detection," in *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2020, vol. 5, pp. 428–434, doi: 10.5220/0008983904280434.

[31] K. M. Kelkar and J. W. Bakal, "Hyper parameter tuning of random forest algorithm for affective learning system," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Aug. 2020, pp. 1192–1195, doi: 10.1109/ICSSIT48917.2020.9214213.

[32] H. Zhang, J. Sun, and Z. Xu, "Adaptive structural hyper-parameter configuration by Q-learning," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2020, pp. 1–8, doi: 10.1109/CEC48606.2020.9185665.

[33] K. Kiatkarun and P. Phunchongharn, "Automatic hyper-parameter tuning for gradient boosting machine," in *2020 1st International Conference on Big Data Analytics and Practices (IBDAP)*, Sep. 2020, pp. 1–6, doi: 10.1109/IBDAP50342.2020.9245609.

[34] S. Putatunda and K. Rama, "A modified bayesian optimization based hyper-parameter tuning approach for extreme gradient boosting," in *2019 Fifteenth International Conference on Information Processing (ICINPRO)*, Dec. 2019, pp. 1–6, doi: 10.1109/ICInPro47689.2019.9092025.

[35] C. Huang, B. Yuan, Y. Li, and X. Yao, "Automatic parameter tuning using bayesian optimization method," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, Jun. 2019, pp. 2090–2097, doi: 10.1109/CEC.2019.8789891.

[36] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: a novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, pp. 1–52, 2018.

[37] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks," *IEEE Access*, vol. 8, pp. 52588–52608, 2020, doi: 10.1109/ACCESS.2020.2981072.

[38] S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: a labeled dataset with malicious and benign IoT network traffic," *Stratosphere Laboratory*, 2020. https://www.stratosphereips.org/datasets-iot23 (accessed Feb. 11, 2022).

BIOGRAPHIES OF AUTHORS

Karthikayini Thavasimani received B.Eng. in Computer Science and Engineering from Avinashilingam University for Women, India and her Master's degree, M.Eng. (Gold Medalist) in Computer Science and Engineering from K.S. Rangasamy College of Engineering, Tiruchengode, Tamil Nadu, India. She is currently working as Data Science Consultant at Capgemini, Bangalore, India and pursuing her part-time Ph.D. in machine learning and big data at RV College of Engineering, Bangalore, Karnataka, India. She has over five years of Academic teaching experience in Computer Science and 1 year of industrial experience in information technology. Her research interests include artificial intelligence, big data, natural language processing, statistical analysis, and optimization techniques. She can be reached at karthikayini@outlook.com.

Nuggehalli Kasturirangan Srinath received his B.E. (Electrical) degree from Bangalore University in 1982, M.Tech (SE and OR) degree from Roorkee University, India in 1986, and Ph.D. from Avinashilingam University, India. He has more than thirty years of teaching experience. He worked as Professor and Dean (Academics) in the Department of Computer Science and Engineering at RV College of Engineering, Bangalore, Karnataka, India. He has authored 7 books, reviewed 4 and published more than 62 journals. He received numerous awards such as "Best Faculty Award" by Cognizant Technologies and Leadership in Academia by Intel. He is subject matter expert in VTU-EDUSAT, Microprocessors and Microcontrollers. He is an expert committee member of the University Grants Commission, India, as Chairman. He can be reached at email: srinathnk@rvce.edu.in.