Enhanced Security Attack Detection and Prevention in 5G Networks using CD-GELU-CNN and FMLRQC with HDFS-ECH-KMEANS

Vinay Gugueoth, guguv@amazon.com

***Abstract*—Security Attacks (SA) refer to a kind of malicious activity in which SA causes information destruction. Existing works have failed to concentrate on various attacks that limit the model's performance. Therefore, this paper presents Cumulative Distribution-GELU-Convolutional Neural Network (CD-GELU-CNN) and Feature Map-based-Lattice Rainbow Quantum Cryptography (FMLRQC) techniques for SA detection and prevention. The proposed system originates from a source system. Firstly, log information is extracted from the 5G AD 2022 dataset. Then, the extracted information is vectorized and classified as attacked or not attacked. After that, the non-attacked data is transferred to the destination system. Then, data security and load balancing are performed. The load-balanced data is then checked in the Intrusion Detection System (IDS) to detect whether the packet is attacked or not. In the IDS, two datasets, namely 5G NDD and 5G SliciNdd datasets, are taken and pre-processed. Common features are extracted from the pre-processed data. Next, feature composition and feature selection are performed to select the best optimal features. Finally, the CD-GELU-CNN detects whether the packet is attacked or not. If the packet is not attacked, it can be transferred to the destination system. Otherwise, a notification is sent to the source system with a feature map to prevent the data. Therefore, the results prove that the proposed model achieved a high accuracy of 98.50%, outperforming prevailing techniques.**

***Index Terms*—5G, Network Security, Machine Learning, Security Attacks, Intrusion Detection Systems.**

I. Introduction

Advancements in mobile wireless communication technology have led to the emergence of numerous applications [1]. Recently, 5G networking has witnessed a multitude of use cases that significantly enhance the achievable data rate. The 5G network aims to achieve specific goals, including ultra-low latency and heterogeneous QoS [2]. Furthermore, 5G possesses a robust capacity for connectivity and has the capability to seamlessly integrate diverse sectors, including smart cities and environmental monitoring [3]. Due to the inherent openness of the 5G network, both authorized and unauthorized users are able to access the network channel [4]. Nevertheless, the rapid advancement of 5G networking also presents certain security issues. Preserving privacy in 5G networking is particularly problematic due to the highly intricate nature of the threats involved [5]. The most prevalent types of Security Attacks (SA) in 5G networks, as identified in reference [6], are impersonation, man-in-the-middle, DDoS attacks, and malicious routing. The network's inadequate security gives rise to numerous issues, including risks to the confidentiality of personal information and the potential for interruption or destruction [7]. So, it is important to protect the communications channel against these types of attacks [8]. Hence, various security mechanisms, including key management, user authentication, and intrusion detection protocols, are employed to protect the networking channel [9]. Many techniques, like Machine Learning (ML), Deep Learning (DL), and Federated Learning (FL), are employed to identify harmful activity in the 5G network [10]. DL approaches, such as Convolutional Neural Network (CNN) and Residual-U-Network (ResUNet), are employed to identify anomalies in the 5G network. In addition, deep learning approaches mitigate the effects of network delay. However, the deep learning model is susceptible to overfitting issues [11]. Deep learning, in conjunction with the Kernelized Support Vector Machine (KSVM) approach, is employed to detect and classify various sorts of signal anomalies, such as jamming. Furthermore, the DL-combined KSVM model significantly improves the security of IDS. However, the DL-combined KSVM model is not capable of processing vast volumes of data, resulting in inaccurate detection [12]. Current research employs FL for data aggregation and Transfer Learning (TL) to create customized detection models to guarantee data security and prevent data leakage. However, the FL model has several limitations [13]. Therefore, the Reinforcement Learning (RL) procedure is employed to identify instances of 5G network attacks occurring at various base stations and servers. Furthermore, RL has computational complexity [14]. However, the above-mentioned models were not fully efficient in detecting the SA over an enhanced 5G network. Therefore, in this paper, an efficient model based on Cumulative Distribution-GELU-Convolutional Neural Network (CD-GELU-CNN) is used to effectively detect the SA and Feature Map-based-Lattice Rainbow Quantum Cryptography (FMLRQC) with Hadoop Distributed File System-based-Entropy Computational Hamming KMeans (HDFS-ECH-KMeans) is used to prevent the model from SA in an enhanced 5G network has been proposed. Existing methods have drawbacks. First, no present technology detects reconnaissance, network reconfiguration, and DDoS attacks simultaneously. Existing data transmission work generally disregards Network Slicing (NS). While several approaches detect these attacks, they often overlook the packet attack tools. Finally, 5G networks transmit sensitive data, making security difficult. The proposed model focuses on achieving key objectives by integrating various techniques. The CD-GELU-CNN detects reconnaissance, network reconfiguration, and DDoS attacks, improving model security. NS

characteristics are extracted during data transit to improve network stability. The model also extracts information from packet attack tools to identify the attacker tool. The secure transmission of data is achieved using the FMLRQC method. The HDFS-based ECH-KMeans approach improves model robustness and effectiveness by efficiently handling large data volumes and preventing traffic congestion.

The rest of the paper is organized as follows: Section II conveys the proposed methodology, Section III explains the results and discussion based on the performance metrics, and Section IV concludes the proposed work along with future recommendations.

II. PROPOSED SA DETECTION AND PREVENTION SYSTEM

In the proposed system, CD-GELU-CNN is used to detect the SA in an enhanced 5G network. The FMLRQC is introduced to secure the data to be transferred. The structural diagram of the proposed model is shown in Figure 1. The

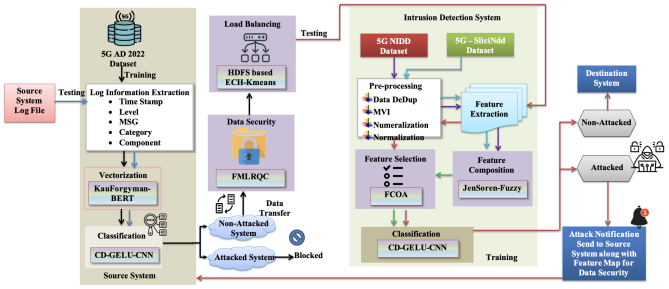


Fig. 1: Structural Design of Proposed Methodology

proposed system transfers the real-time data from the source system to the destination system. The process of the proposed system is explained below,

A. Source System

Primarily, the proposed system initiates from the source system. Here, the source system is checked to identify whether it is attacked by reconnaissance and network reconfiguration attacks or not.

1) Training: The source system is trained based on processes such as log information extraction, vectorization, and classification.

i) Dataset: Here, the 5G AD 2022 dataset is taken to check whether the source system is attacked or not. The 5G AD 2022 dataset contains a total k number of data. It is expressed as, $\vartheta_m^{\bullet} = \{\vartheta_1^{\bullet}, \vartheta_2^{\bullet}, ..., \vartheta_{\Lambda}^{\bullet}\}$. Where, $\vartheta_m^{\bullet}$ represents the 5G AD 2022 dataset data, $m = 1, 2, ..., \Lambda$.

ii) Log Information Extraction: Next, the log information, such as time, level, msg, category, and component, are extracted from the dataset. The extracted log information is helpful to identify the SA. The extracted log information ζ^H can be equated as, $\zeta^H = [\zeta^1, \zeta^2, ..., \zeta^{\Gamma}]$. Here, ζ^{Γ} denotes the total number of ζ^H, $H = 1, 2, ..., \Gamma$.

iii) Vectorization: After that, the ζ^H is converted into vector format to get some distinct log information out of the text. The vectorization is performed using the KauForgyman-Bidirectional Encoder Representation from Transformers (KauForgyman-BERT) algorithm, which easily trains the classifier. The BERT model makes the smaller and more defined natural language processing tasks easier. But, in the BERT model, the weights are created randomly from the normal distribution at the start of the training process, which makes the model worse. So, the performance of the model was degraded. To overcome these problems, KauForgyman-based weights are included with the BERT. The ζ^H is taken as input for the vectorization process.

- Positional Encoding: Firstly, each input embedding consists of three embeddings such as token embedding, segmentation embedding, and positional embedding. In token embedding $\wp$, $\wp_{start}$ token is added at the start of the first sentence, and $\wp_{end}$ token is added at the end of the sentence. In between $\wp_{start}$ and $\wp_{end}$, the words are masked, which can be denoted as $\wp_{masked}$. It can be expressed as, $\wp(\Im_c) = \{\wp_{start}, ..., \wp_{masked}, ..., \wp_{end}\}$. Then, the segmentation embedding is performed to identify the sentences. After that, the positional embedding shows the position of each token in a sentence. The position can be identified by using the below formula,

$$P_{(ps,2h)} = \sin\left[\frac{ps}{10000^{\frac{2h}{D_{model}}}}\right] \quad (1)$$

$$P_{(ps,2h+1)} = \cos\left[\frac{ps}{10000^{\frac{2h}{D_{model}}}}\right] \quad (2)$$

Where ps denotes the position of the word in the sentence, h represents the dimension of the word vector, and D_{model} is the dimension of the total word vector. The obtained position information τ_u is denoted as, $\tau_u = [\tau_1, \tau_2, ..., \tau_{\infty}]$ where $u = 1, 2, ..., \infty$. Next, the vector expression is calculated in the following steps.

- Compute word vector and positional encoding: Then, each word in the vector table and τ_u is added to get the vector expression, which can be defined as $\cup = \wp + \tau_u$. Where $\cup$ is the computed vector format.
- Self-Attention mechanism: Each word is associated with all other words in the sentence. So, a multi-head self-attention $\cup_{att}$ mechanism is calculated to extract the multiple semantics. It can be expressed as

$$\cup_{att} = Softmax\left[\frac{FU^T}{\sqrt{D_u}}\right] S \quad (3)$$

Where FU^T represents the attention matrix and D_u indicates the dot product of the matrix. Here, F, U, and S are the three matrices which are formed while performing the linear mapping of $\cup$. Then, the KauForgyman is added with BERT to update the weights, which improves the performance. Also, the KauForgyman helps to keep the variance of activations and

gradients consistent across layers. It can be formulated as,

$$F = L(\cup) = \cup \bullet \frac{\parallel W_{F_i} - W_{F_j} \parallel}{\sum F_{ij}} \tag{4}$$

$$U = L(\cup) = \cup \bullet \frac{\parallel W_{U_i} - W_{U_j} \parallel}{\sum U_{ij}} \tag{5}$$

$$S = L(\cup) = \cup \bullet \frac{\parallel W_{S_i} - W_{S_j} \parallel}{\sum S_{ij}} \tag{6}$$

where, W_F, W_U, and W_S are the weights of the F, U, and S matrices, correspondingly and i and j denote the rows and columns.

- Residual connection and standardization: Next, $\Im_c$ and $\cup_{att}$ are added to make better residual connections. It can be expressed as $\cup_{att} = \Im_c + \cup_{att}$. To decrease the execution time, the added outcomes are normalized to make the $\cup_{att}$ as standardized. The normalized outcome is expressed as,

$$\Theta_{out} = LayerNorm\left[\alpha \bullet \frac{\Im_c - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta\right] \tag{7}$$

here, α and β are the normalization constants, μ and σ represent the mean and variant, respectively, and ε is used to prevent the denominator from being 0. Finally, the obtained vectorized text Ω^o is expressed as, $\Omega^o = (\Omega^1, \Omega^2, ..., \Omega^\ell)$. Here, Ω^1 and Ω^ℓ indicate the first and last Ω^o.

iv) Classification: After that, the classifier detects whether the Ω^o is attacked or not attacked. Here, CD-GELU-CNN is used to efficiently identify the SA. CNN consumes less energy and obtains faster results. However, CNN is affected by slow convergence and low learning efficiency problems. To overcome this issue, the CD-GELU activation function is included with CNN. The CD-GELU-CNN consists of an Input layer, a Convolutional layer, a Pooling layer, a Fully Connected (FC) layer, and an Output layer with CD-GELU activation function.

- Input layer: Initially, Ω^o is fed to the input layer. Then, the input layer data are subjected to the convolutional layer for further processing. The total $\Diamond$ number of input layer data can be expressed as, $A_\delta(\Omega^o) = \{A_1, A_2, ..., A_\Diamond\}$. Where A_δ indicates the input layer data.
- Convolutional layer: Next, the convolutional layer convolutes the A_δ and saves in different channels. The convolutional layer extracts the features from the A_δ and also reduces the number of features. The convolutional layer generates the feature map G, which can be expressed as $G = w_l A_\delta + b_l$. Where w_l and b_l indicate the weight vector and bias terms with $l = 1, 2, ..., a$ number of weights and biases.
- Pooling layer: Pooling layers are used to reduce the size of the G. Also, it makes the computation faster. Each feature map of a pooling layer is attached to its corresponding feature map in the convolutional layer. The pooling function for each G is denoted as $p_k = pool(G)$. Where p_k indicates the pooling function.
- FC layer: FC layers predict the class of the input data corresponding to the features extracted from the previous layers. FC layer takes all the previous layer outputs and connects all the layers into every single layer. The FC layer is determined as,

$$\psi_{Z'}(p_k) = w_l + \left[\sum_{k=1}^{n} p_k\right] + b_l \tag{8}$$

Where, $\psi_{Z'}$ represents the FC layer with $Z' = 1, 2, ..., s$.

- Output layer: The output layer O is the last layer of the CD-GELU-CNN. The CD-GELU activation function is added with CNN to overcome the slow convergence and low learning efficiency problem. CD-GELU ξ has a smooth and more continuous shape than other functions, which can make it very effective at learning complex patterns. It efficiently avoids the slow convergence problem. The ξ can be defined as

$$\xi = 0.5P'(w_l) \bullet \left[1 + \frac{2}{\sqrt{b_l}} \int_0^{\frac{P'(w_l)}{2}} e^{-l^2}\, dl\right] \tag{9}$$

Where P' indicates the probability. The output of ξ is subjected to the output layer in order to detect whether the Ω^o is attacked or not. It can be expressed as, $O(\xi) \Rightarrow (\Pi, \Re)$. Finally, the CD-GELU-CNN delivers the results, such as Attacked Π or Non-attacked $\Re$. The CD-GELU-CNN activation function, followed by the output layer, generates the classification results, such as attacked or non-attacked.

2) Testing: The data in the source system log file is tested in the above-trained CD-GELU-CNN. The data in the source system log file $S'_{e'}$ can be expressed as, $S'_{e'} = (S'_1, S'_2, .., S'_{h'})$. Here $e' = 1, 2, ..., h'$. The log information is extracted from the $S'_{e'}$. Then, vectorized and classified, such as attacked Π' or not-attacked $\Re''$. If the data in the source system log file is non-attacked, then the non-attacked data $\Re''$ is transferred. Otherwise, the Π' is blocked.

B. Data Transfer

After that, the non-attacked system's data $\Re''$ is transferred to the destination system. The data is secured at this point, and load balancing is performed to avoid the traffic congestion of large amounts of data.

1) Data Security: Then, the $S'_{e'}$ data is secured by using FMLRQC. Lattice Quantum Cryptography has the ability to solve complex computing problems quickly. But, it is affected by side-channel attacks. So, a rainbow signature scheme is added with LQC to avoid side-channel attacks. Initially, the source system $S'_{e'}$ and the destination system Rr generate two sets of keys based on the feature map values G of the CD-GELU-CNN. The coefficients ($\alpha''_{ij}, \beta''_{ij}$ and γ''_i) for the feature map are discovered based on the rainbow signature scheme.

So, most of the possibility of attacks is avoided. The rainbow signature scheme is defined as

$$\Phi' = \sum_{ij \in G} \alpha''_{ij} + \sum_{ij \in G} \beta''_{ij} + \sum_{i \in G} \gamma''_{i} + G \tag{10}$$

The public key is denoted as $\aleph_{pu}$ and $\aleph'_{pu}$ and the private key is denoted as $\nabla_{pr}, \nabla'_{pr}$, which can be expressed as,

$$S'_{e'} \rightarrow (\aleph_{pu}, \nabla_{pr}) \ \ and \ \ Rr \rightarrow (\aleph'_{pu}, \nabla'_{pr}) \tag{11}$$

While transferring the data to the destination system, the $\Re''$ is encrypted by using the public key $\aleph_{pu}$ of the source system. The encrypted data $\chi_{\infty'}$ with the ω' number of data can be expressed as, $\chi_{\infty'} = En(\Re'', \aleph_{pu})$. Where En represents the encryption cipher and ∞' denotes the total number of encrypted data. At the destination system side, the data is decrypted by using the private key of the source system. The decrypted data $\Re''$ can be expressed as, $\Re'' = De(\chi_{\infty'}, \nabla_{pu})$. Here, De represents the decryption cipher.

2) Load Balancing: After that, the encrypted data $\chi_{\infty'}$ is load-balanced to handle the large amounts of data from traffic congestion. Here, the HDFS-based ECH-KMeans algorithm is utilized to load balance the data. The HDFS is used to avoid the high response time of the load balancing technique. KMeans algorithm is easy to understand and implement. However, the KMeans algorithm randomly initializes the cluster centroids at the beginning. So, the final results are varied depending on these initializations, which causes problems in KMeans. So, ECH distance is calculated in the KMeans to improve the clustering process. HDFS is utilized to handle a large number of data. HDFS performs both mapping and reducing. Then, the load-balanced data is subjected to the IDS. The ECH-KMeans algorithm maps the transferred data based on network traffic features, such as bandwidth, latency, and various protocols. The $\chi_{\infty'}$ are primarily initialized and given as input to the mapping process. The mapping process decides the number of clusters by mapping the transferred data based on network traffic features, such as bandwidth, latency, and various protocols.

1) The centroids E_{ab} with the total ∞ number of centroids are selected randomly from $\chi_{\infty'}$ and can be expressed as, $E_{ab} = \{E_1, E_2, ..., E_\infty\}$. Here, E_2 indicates the second centroid.
2) Then, the ECH distance d is calculated instead of Euclidean distance to improve the clustering process. ECH finds the distance between E_{ab} and $\chi_{\infty'}$. ECH is helpful in finding the closest centroid. It can be denoted as,

$$d(\chi_{\infty'}, E_{ab}) = - \sum_{\infty'=1, ab=1}^{\omega', \infty} \frac{}{\sqrt{\parallel P''(E_{ab})^2 . \log P''(E_{ab}) - P''(\chi_{\infty'})^2 . \log P''(\chi_{\infty'}) \parallel}} \tag{12}$$

Where P'' indicates the probability.

3) Next, the average for all data that belongs to each cluster is calculated to compute a new centroid for each cluster.

$$\zeta'_{\ell'} = \frac{\sum_{\infty'=1, ab=1}^{A', \infty} t_{\infty' ab} \chi_{\infty'}}{\sum_{\infty'=1, ab=1}^{A', \infty} t_{\infty' ab}} \tag{13}$$

where $\zeta'_{\ell'}$ is the new centroid of the cluster and $t_{\infty' ab}$ is the average of data. After that, each data is reassigned to the new closest centroid of each cluster. Next, step 3 is followed to find a new centroid. These steps are continued until the best clusters are discovered. Finally, the obtained mapped data B_∂ with the total $\angle$ number of data is expressed as, $B_\partial = (B_1, B_2, ..., B_\angle)$. Where B_1 is the first mapped data. Next, the mapped data B_∂ are fed to the reducer that collects the mapped data and joins those in a single output file. Finally, the outcome of the reducer is considered as load-balanced data, which is stored in HDFS. The load-balanced data $\upsilon_{b'}$ is derived as, $\upsilon_{b'} = \{\upsilon_1, \upsilon_2, ..., \upsilon_{bc}\}$. Here, υ_{bc} represents the total number of load-balanced data and $b' = 1, 2, ..., bc$.

C. IDS

Then, the load-balanced data $\upsilon_{b'}$ is subjected to the IDS. Here, IDS checks whether the load-balanced data is attacked or not.

1) Training: The IDS is trained based on the following processes such as pre-processing, feature extraction, feature composition, feature selection, and classification. It can be explained below,

i) Dataset: In IDS, two datasets, namely 5G NIDD and 5G SliciNdd, are taken to train the classifier. a) 5G NIDD: The 5G NIDD dataset contains the NS features. The total $\perp$ number of data in the 5G NIDD dataset $\phi_\otimes$ can be expressed as, $\phi_\otimes = [\phi_1, \phi_2, ...\phi_\perp]$. b) 5G SliciNdd: The 5G SliciNdd dataset contains the attack type and attack tool type features. The total a' number of data in the 5G SliciNdd dataset η_z can be defined as, $\eta_z = [\eta_1, \eta_2, ..., \eta_{a'}]$. Here, $\eta_{a'}$ indicates the total number of data.
ii) Pre-Processing: After that, the dataset data $\phi_\otimes$ and η_z are pre-processed individually to improve the quality of the data. Firstly, the duplicate data that exists in the rows and columns is eliminated. Then, the missing values are replaced randomly, which can be denoted as v. After that, numeralization is performed to convert the string data into numerical values. The numeralized data can be determined as, $\beta'_M(\nu) = (\beta'_1 + \beta'_2 + ...\beta'_{c'})$ where $M = 1, 2, ..., c'$. Here, β'_M represents the numeralized data. Then, β'_M is normalized with respect to minimum $\min\beta'_M$ and maximum $\max\beta'_M$ values. All the numbers are converted into a normalized range of 0 or 1. The normalized data N_φ can be expressed as,

$$N_\varphi(\beta'_M) = \frac{\beta'_M - min(\beta'_M)}{max(\beta'_M) - min(\beta'_M)} \tag{14}$$

Finally, the pre-processed data $\tau'_{B'}$ with $B' = 1, 2, .., i'$ for the dataset data $\phi_\otimes$ can be defined as, $\tau_{B'}(\phi_\otimes) =$

$\{\tau_1, \tau_2, ...\tau_{i'}\}$. The pre-processed data $\lambda_{j'}$ for the dataset data η_z can be expressed as, $\lambda_{j'}(\eta_z) = \{\lambda_1, \lambda_2, ..., \lambda_{\Gamma'}\}$. Where, $\lambda_{\Gamma'}$ is the total number of pre-processed η_z.

iii) Feature Extraction: After pre-processing, the features, such as duration, runtime, mean, protocol, label, attack type, attack tool, state, loss, load, offset, and sequence are extracted from the $\lambda_{\gamma}^{\oplus}(\tau'_{B'})$. The extracted features $\lambda_{\gamma}^{\oplus}$ expressed as, $\lambda_{\gamma}^{\oplus}(\tau'_{B'}) = \{\lambda_1^{\oplus}, \lambda_2^{\oplus}, ..., \lambda_i^{\oplus}\}$. Where $\lambda_i^{\oplus}$ denotes the total number of extracted features, $\gamma = 1, 2, .., \imath$. Then, features such as UniqueID, SynAck, AckDat, loss, load, offset, cause, runtime, protocol, duration, and sequence are extracted from the $\lambda_{j'}$. The extracted features $\Psi_L^{\oplus}$ defined as, $\Psi_L^{\oplus}(\lambda_{j'}) = \{\Psi_1^{\oplus}, \Psi_2^{\oplus}, .., \Psi_{\sigma'}^{\oplus}\}$. Where, $\Psi_1^{\oplus}$ denotes the first extracted feature, $L = 1, 2, .., \sigma'$.

iv) Feature Composition: After that, feature composition is performed to merge the NS classes into attack type and tool classes by calculating the similarity between $\lambda_{\gamma}^{\oplus}$ and $\Psi_L^{\oplus}$. This process improves the efficiency of the classification. The implementation of a fuzzy logic system is easy, simple, and flexible. But, fuzzy has tuning difficulties in generating the set of rules. So, a JenSoren similarity is calculated using the fuzzy rules to provide better performance. Primarily, the fuzzy rule base takes the extracted features of the two datasets as inputs. JenSoren Fuzzy uses the definite fuzzy If-Then rules along with the AND or OR operator to create the rules. The fuzzy rule is generated based on the JenSoren similarity calculation. Firstly, JenSoren similarity is calculated, which can be expressed as,

$$\in = \sum Pp(\lambda_{\gamma}^{\oplus}) \frac{2 * |Pp(\lambda_{\gamma}^{\oplus}) \cap Pp(\Psi_L^{\oplus})|}{|Qq(\lambda_{\gamma}^{\oplus})| + |Qq(\Psi_L^{\oplus})|} \tag{15}$$

Here, Pp and Qq indicate the probability distribution. Then, the fuzzy rules find the similarity between the $\lambda_{\gamma}^{\oplus}$ and $\Psi_L^{\oplus}$; also merge the features of the two datasets and store them into one dataset. The set of fuzzy rules can be defined as,

$$if(\lambda_1^{\oplus} \in \Psi_1^{\oplus})\&\&...\&\&(\lambda_c^{\oplus} \in \Psi_c^{\oplus})\ then\ X \tag{16}$$

$$if(\lambda_1^{\oplus} \notin \Psi_1^{\oplus})\&\&...\&\&(\lambda_c^{\oplus} \notin \Psi_c^{\oplus})\ then\ K \tag{17}$$

Where $\notin$ represents the more distant value, $\in$ indicates the nearest value, X denotes the two datasets merged into one dataset, and K indicates the two datasets not merged into one dataset. Next, the rule generation unit performs interference operations, such as fuzzification and defuzzification, based on the fuzzy If-Then rule. A fuzzification ϑ' interference unit converts the crisp data ϕ'_{xy} into fuzzy data χ'_{pq}, $\vartheta' = (\phi'_{xy} \rightarrow \chi'_{pq})$. Then, a Defuzzification Ω' interference unit performs the inverse operation of converting the fuzzy data into crisp data. $\Omega' = (\chi'_{pq} \rightarrow \phi'_{xy})$. This JenSoren-Fuzzy merges similar feature values into the 5G SliciNdd dataset. Finally, the merged features V^e in the 5G NIDD dataset are expressed as, $V^e = \{V^1, V^2, ..., V^{d'}\}$ where $e = 1, 2, .., d'$. Where $V^{d'}$ is the total number of merged features.

v) Feature Selection: Then, the optimal features are selected from the V^e to perform efficient classification. FOA has the ability to solve complex problems and produce optimal solutions by using the three-stage action settlement. Due to the random initialization factor, the FOA has low convergence speed and steady-state oscillations. So, the Crest factor is included in the position updation phase. Primarily, the population of the falcons is initialized. The initialized population $J^{\delta'}$ is considered as the merged features, which is expressed as $J^{\delta'} = [J^1, J^2, ..., J^Y]$. Where, J^Y is the total number of falcons and $\delta' = 1, 2, .., Y$. Next, the velocity and position of the falcons are selected randomly in a D' dimensional space based on upper bound $upper$ and lower bound $lower$. The position of the falcons is selected by the population and D' dimension. The speed of the falcons is expressed as,

$$\nu_{max} = 0.1 * (upper - lower) \tag{18}$$

$$\nu_{min} = -\nu_{max} \tag{19}$$

Where ν_{max} is the high speed and ν_{min} is the low speed. Then, the fitness f is calculated by assigning the classification accuracy as the maximum, which is equated by,

$$f = max(Acc) * f(J^{\delta'}) \tag{20}$$

$$J_{best}^{\delta'} = Best(f) \tag{21}$$

The best value obtained for f corresponds to the best candidate solution $J_{best}^{\delta'}$. Then, the falcon moves to seek the prey, so the position of the falcon is updated. The crest factor w is calculated in the position updation I_{it+1} to increase the convergence speed. It can be defined as,

$$w = \frac{I_{it}}{\sqrt{\frac{1}{I_{it}} \sum_{it=1}^{max} I_{it}^2}} \tag{22}$$

$$I_{it+1} = w + \nu_{it} + q(I_{best}, I_{it} + r(J_{best}^{\delta'}, I_{it})) \tag{23}$$

Here, I_{it} indicates the current position of the falcon, ν_{it} is the current velocity, q is the cognitive rate, and is the social constant. The pair (pQ', pT') of dive probability Q' and awareness probability T' are generated randomly for each falcon. If the pQ' is bigger than Q', then one target is chosen I_{chosen} by the falcon. Then, the falcons step forward to hunting. The new updated suitable position can be expressed as,

$$I'_{it+1} = I_{it} + |I_{chosen} - I'_{it}|.e^{bbtr} \cos(2\pi t) \tag{24}$$

Where bb indicates the fixed number and indicates the arbitrary number within a range of [1,-1]. If the Q' is bigger than pQ', then the score function of the preferred prey and the score function of the falcon are compared. If the prey is appropriate, then this dive probability process is followed by the falcon, which is formulated as,

$$I_{it+1} = I_{it} + \nu_{it+1} + Z.\, ran(I_{chosen} - I_{it}) \tag{25}$$

Where ν_{it+1} is the new velocity, ran indicates the randomly chosen, and Z is the following constant. Otherwise, the falcon continuously flew based on its best position, which is determined as,

$$I_{it+1} = I_{it} + \nu_{it+1} + q.\, ran(I_{best}, I_{it}) \qquad (26)$$

Then, the Falcons score function is calculated, and the best fitness values are calculated. The above process is continued until the best optimal solutions are obtained. Finally, the selected features $\Re'_{m'}$ with the total μ' number of features are expressed as $\Re'_{m'} = [\Re_1 + \Re_2 + ... + \Re_{\mu'}]$. Here, $\Re_1$ represents the first $\Re'_{m'}$. Next, the $\Re'_{m'}$ are subjected to the classifier to detect the SA. It can be explained in the following section.

vi) Classification: Finally, the selected features are fed to the classifier that identifies whether the transferred data is attacked or non-attacked. Here, CD-GELU-CNN is utilized to detect the SA of the transferred data. The CD-GELU-CNN process is explained in section III.A.1.iv. This classifier also detects the SA based on the above-explained process. It delivers the classification results, such as whether the $S'_{e'}$ is attacked or non-attacked.

2) Testing: The load-balanced data $\upsilon_{b'}$ is tested by using the above-trained classifier. This executes processes such as feature value extraction, pre-processing, feature selection, and classification. Finally, the IDS detects whether the decrypted data $S'_{e'}$ is attacked or non-attacked. If the $S'_{e'}$ is non-attacked, then it is transferred to the destination system. Otherwise, a notification alert is sent to the source system along with a feature map to secure the data.

III. RESULTS AND DISCUSSION

In this section, the performance of the proposed model is evaluated by comparing it with existing methods based on the performance metrics. The proposed model is implemented using the PYTHON programming language. The suggested system utilized 3 datasets: 5G AD 2022 [15], 5G NIDD [16], and 5G SliciNdd [17]. The 5G AD 2022 dataset contains extensively detailed log files. The 5G NIDD dataset has 14456 NS characteristics. The 5G SliciNdd dataset includes 1048575 attack types and tool attributes. Datasets are gathered from publically available sources, cited in the reference section. Out of the total data, 80% is used for training and 20% for testing. Initially, the suggested model is compared to existing approaches like CNN, RNN, DBN, and DNN for performance analysis. This article validates the performance of the proposed CD-GELU-CNN on the 5G AD 2022 dataset. The precision, recall, f-measure, accuracy, sensitivity, and specificity of the proposed CD-GELU-CNN are compared with prevailing techniques, as illustrated in Fig. 2. The CD-GELU-CNN showed high precision (98.32%), recall (98.85%), f-measure (98.36%), accuracy (98.41%), sensitivity (98.14%), and specificity (98.25%). In contrast, other studies found lower mean accuracy (93.85%), sensitivity (95.65%), specificity (94.54%), precision (93.62%), recall (95.81%), and f-measure (91.87%). The CD-GELU activation function gives the CD-GELU-CNN great accuracy. Thus, the CD-GELU-CNN outperforms other methods.

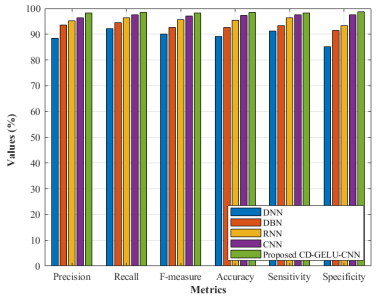


Fig. 2: Comparison of the proposed model based on performance metrics

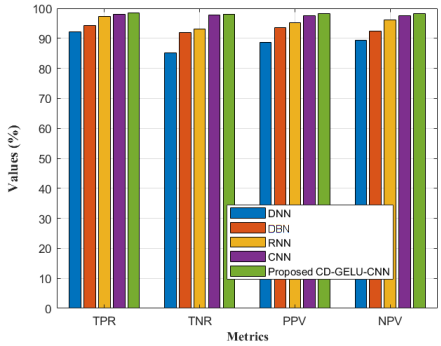
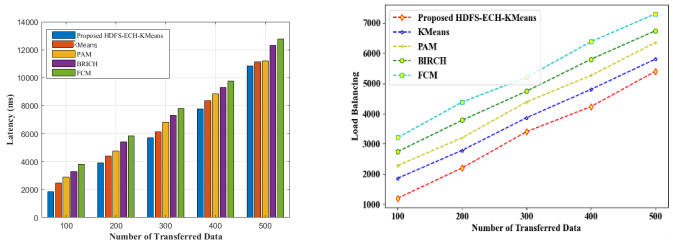


Fig. 3: Graphical analysis of the proposed CD-GELU-CNN

Next, Fig. 3 validates the 5G NIDD and 5G SliciNdd datasets by graphically analyzing the proposed model's True Positive Rate (TPR), True Negative Rate (TNR), Positive Predictive Value (PPV), and Negative Predictive Value (NPV) alongside other methodologies. The suggested model performs better than the RNN technique, which had lower rates of TPR (96.35%), TNR (92.15%), PPV (94.87%), and NPV (94.37%) and a larger training time of 47125ms. These results prove the CD-GELU-CNN model's ability to avoid slow convergence and low learning efficiency. Additionally, the model improves classification above current methods. In Fig. 4, the proposed HDFS-ECH-KMeans is compared to KMeans, Partition Around Medoids (PAM), Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), and Fuzzy-CMeans (FCM) in terms of latency and load balancing. HDFS-ECH-KMeans performed best with 10847ms latency and 5402 load balancing for 500 transmitted data. Other methods have an average latency of 11855ms and load balancing of 6553.25 for the same data amount. Thus, the proposed load-balancing model reduced traffic congestion more than existing strategies. Finally, to prove its efficacy, the model is compared to others.

Table I compares the proposed framework to current works. The CD-GELU-CNN detects SA in an improved 5G network in the proposed study. Other efforts, notably Multi Kernel-KMeans (MKKM), misdetected SA due to low accuracy, precision, and f-measure values of 95.60%, 88.24%, and 89.11%.



(a) (b)

Fig. 4: Performance estimation of HDFS-ECH-KMeans regarding (a) Latency and (b) Load balancing

TABLE I: Comparative Analysis of the proposed model

Study	Method	Dataset Used	Accuracy (%)	Precision (%)	F-measure(%)
Proposed model	**CD-GELU-CNN**	5G AD 2022, 5G-NIDD and 5G-SliciNdd	**98.50**	**98.58**	**98.96**
[18]	**CNN**	**Malimg**	**99**	**92**	**88**
[19]	**MKKM**	**AWID**	**95.60**	**88.24**	**89.11**
[20]	**KBNN**	**CICDDoS 2019**	**94**	**91.22**	**94.47**
[21]	**MEOADL**	**Benchmark dataset**	**97.60**	**97.58**	**97.55**
[22]	**ML**	**Unified dataset**	**98.27**	**98.81**	**98.54**

The Kalman Backpropagation-NeuralNetwork (KBNN) has 94% accuracy, while the proposed model had 98.50%. KBNN performance suffers. Other methods, including CNN, ML, and MEOADL, also had low f-measures. Therefore, the proposed approach successfully achieved accurate results in detecting the SA.

IV. Conclusion and Future Work

The paper proposes an efficient framework for detecting SA and preventing data compromise from such attacks. This is achieved by utilizing the CD-GELU-CNN and FMLRQC with HDFSECH-KMeans. In the proposed model, the source system utilizes the 5G AD 2022 dataset, executing processes such as log information extraction, vectorization, and classification. The transferred data are then secured and load-balanced. Subsequently, the Intrusion Detection System (IDS) employs the 5G NIDD and 5G SliciNdd datasets, undertaking preprocessing, feature extraction, feature decomposition, feature selection, and classification. After completing all processes, the proposed work attains a high accuracy of 98.41% for the 5G AD 2022 dataset and 98.59% for the 5G NIDD and 5G SliciNdd datasets. This demonstrates the efficient classification capability of the proposed method.The research methodology focuses solely on NS features, attack types, and tool features for classification based on CD-GELU-CNN. Future research will concentrate on energy efficiency and network discontinuities in an enhanced 5G network.

References

[1] Y. Sun, Z. Tian, M. Li, C. Zhu, and N. Guizani, "Automated attack and defense framework toward 5g security," *IEEE Network*, vol. 34, no. 5, pp. 247–253, 2020.

[2] S. Sullivan, A. Brighente, S. A. Kumar, and M. Conti, "5g security challenges and solutions: a review by osi layers," *IEEE Access*, vol. 9, pp. 116 294–116 314, 2021.

[3] A. S. Abdalla, K. Powell, V. Marojevic, and G. Geraci, "Uav-assisted attack prevention, detection, and recovery of 5g networks," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 40–47, 2020.

[4] A. Dutta and E. Hammad, "5g security challenges and opportunities: a system approach. in 2020 ieee 3rd 5g world forum (5gwf)(pp. 109-114)," 2020.

[5] M. Wazid, A. K. Das, S. Shetty, P. Gope, and J. J. Rodrigues, "Security in 5g-enabled internet of things communication: issues, challenges, and future research roadmap," *IEEE Access*, vol. 9, pp. 4466–4489, 2020.

[6] Y. Kim, J. G. Park, and J.-H. Lee, "Security threats in 5g edge computing environments," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2020, pp. 905–907.

[7] S. Kwon, S. Park, H. Cho, Y. Park, D. Kim, and K. Yim, "Towards 5g-based iot security analysis against vo5g eavesdropping," *Computing*, vol. 103, pp. 425–447, 2021.

[8] Y. Li, S. Liu, Z. Yan, and R. H. Deng, "Secure 5g positioning with truth discovery, attack detection, and tracing," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 22 220–22 229, 2021.

[9] T. Madi, H. A. Alameddine, M. Pourzandi, and A. Boukhtouta, "Nfv security survey in 5g networks: A three-dimensional threat taxonomy," *Computer Networks*, vol. 197, p. 108288, 2021.

[10] A. Boualouache and T. Engel, "A survey on machine learning-based misbehavior detection systems for 5g and beyond vehicular networks. arxiv 2022," *arXiv preprint arXiv:2201.10500*.

[11] M. Doan and Z. Zhang, "Deep learning in 5g wireless networks-anomaly detections," in *2020 29th Wireless and Optical Communications Conference (WOCC)*. IEEE, 2020, pp. 1–6.

[12] M. Hachimi, G. Kaddoum, G. Gagnon, and P. Illy, "Multi-stage jamming attacks detection using deep learning combined with kernelized support vector machine in 5g cloud radio access networks," in *2020 international symposium on networks, computers and communications (ISNCC)*. IEEE, 2020, pp. 1–5.

[13] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, "Iotdefender: A federated transfer learning intrusion detection framework for 5g iot," in *2020 IEEE 14th international conference on big data science and engineering (BigDataSE)*. IEEE, 2020, pp. 88–95.

[14] H. Sedjelmaci, "Cooperative attacks detection based on artificial intelligence system for 5g networks," *Computers & Electrical Engineering*, vol. 91, p. 107045, 2021.

[15] C. Coldwell, D. Conger, E. Goodell, B. Jacobson, B. Petersen, D. Spencer, M. Anderson, and M. Sgambati, "5GAD-2022," Aug. 2022. [Online]. Available: https://github.com/IdahoLabResearch/5GAD/

[16] S. Samarakoon, Y. Siriwardhana, P. Porambage, M. Liyanage, S.-Y. Chang, J. Kim, J. Kim, and M. Ylianttila, "5g-nidd: A comprehensive network intrusion detection dataset generated over 5g wireless network," 2022. [Online]. Available: https://dx.doi.org/10.21227/xtep-hv36

[17] K. TAHORI, "5g-slicindd. figshare. dataset," 2023. [Online]. Available: https://doi.org/10.6084/m9.figshare.24446515.v1

[18] A. Anand, S. Rani, D. Anand, H. M. Aljahdali, and D. Kerr, "An efficient cnn-based deep learning model to detect malware attacks (cnn-dma) in 5g-iot healthcare applications," *Sensors*, vol. 21, no. 19, p. 6346, 2021.

[19] N. Hu, Z. Tian, H. Lu, X. Du, and M. Guizani, "A multiple-kernel clustering based intrusion detection scheme for 5g and iot networks," *International Journal of Machine Learning and Cybernetics*, pp. 1–16, 2021.

[20] M. Almiani, A. AbuGhazleh, Y. Jararweh, and A. Razaque, "Ddos detection in 5g-enabled iot networks using deep kalman backpropagation neural network," *International Journal of Machine Learning and Cybernetics*, vol. 12, pp. 3337–3349, 2021.

[21] M. Aljebreen, F. S. Alrayes, M. Maray, S. S. Aljameel, A. S. Salama, and A. Motwakel, "Modified equilibrium optimization algorithm with deep learning-based ddos attack classification in 5g networks," *IEEE Access*, 2023.

[22] S. J. Rani, I. Ioannou, P. Nagaradjane, C. Christophorou, V. Vassiliou, H. Yarramsetti, S. Shridhar, L. M. Balaji, and A. Pitsillides, "A novel deep hierarchical machine learning approach for identification of known and unknown multiple security attacks in a d2d communications network," *IEEE Access*, 2023.