ORIGINAL ARTICLE

Intelligent deep learning framework for cache pollution threatening detection using named data networking

S. Sethu[1] · A. Manikandan[1]

Received: 6 February 2025 / Accepted: 21 August 2025

Abstract
Malicious management of the caching process disrupts data retrieval, causing an unavoidable threat known as a cache pollution attack (CPA), which reduces network performance, increases information retrieval time, and decreases the cache hit rate. Due to the importance of CPA detection in NDN, this study introduces the enhanced deep learning (EDL) model to improve data integrity and security. The EDL model integrates the idealogy of recurrent neural networks and a memetic optimization algorithm to eliminate malicious user participation in networks. During the analysis, the CICIDS 2017 dataset was utilized due to its high-dimensional data and coverage of diverse attacks. The gathered details are processed by traffic preprocessing blocks that remove irrelevant, missing, and redundant information by considering specific filtering conditions. Then, the normalization process is applied to mitigate the overfitting issue and reduce computational complexity. The normalized inputs are fed into the recurrent layer, which identifies the relationship between features such as temporal patterns, cache hit rates, and frequency of access requests. According to the relationship, the malicious and legitimate users are identified with maximum accuracy. In addition, memetic operators such as selection, crossover, and mutation parameters are utilized to balance the stability between the features. The effective selection parameter ensures optimal results while managing the cache in NDN, achieving a 3.2% error rate and a cache hit rate of 98% to 99%.

Keywords Named data networking · Cache pollution attack · Traffic processing · Memetic optimization · Recurrent layer · Stability

1 Preliminary analysis

The data-centric networking concept, such as named data networking (NDN) [1], is a concept in which data is accessed by initiating a name request instead of its location. Caching [2, 3] is a key feature of NDN that enables routers to store data packets temporarily, thereby enhancing efficiency and minimizing transmission redundancy. This caching process is more vulnerable to NDN because cache pollution attacks (CPAs) [4] are created, which compromise the efficacy and integrity of the caching layer. The CPA occurred while malicious users were attempting to transmit several requests for unfamiliar content, which resulted in the loss of legitimate user caches [5]. Therefore, the CPA maximizes the retrieval delay, and NDN resource exhaustion also affects typical caching characteristics, minimizing network performance by creating bottlenecks [6]. The NDN faces several types of CPA attacks, including cache flooding, cache poisoning, and interest flooding attacks. In cache poisoning [7], the attacker inserts malicious data into the cache, thereby altering the original information. If a trusted user accesses the information and retrieves false or invalid data, it affects system reliability. Then, cache flooding [8] occurs, in which attackers demand a high volume of content, causing legitimate users to fail to access high-quality

https://doi.org/10.1007/s00521-025-11620-9
Published online: 17 September 2025

information and thereby minimizing network performance and cache hit rates. The last type is interest flooding [9], in which the attacker indirectly creates cache pollution by crushing the network with the help of more interest packets. The type of CPA [10, 11] is illustrated in Fig. 1.

The different types of CPA create various impacts [12], such as reduced cache efficiency, performance degradation, and service denial due to frequent cache exclusions and repeated uncached content [13, 14]. Various mitigation techniques are utilized to identify and mitigate CPA in NDN. The mitigation techniques are described in Fig. 2.

Security threats involving the NDN cache system underscore the necessity of effective security mechanisms for in-network cache storage while maintaining efficiency. By eliminating such flaws, NDN is a strong challenger for the future of scalable and secure data-centric networking.

1.1 Concern clarification

Cache pollution attack (CPA) identification is a complex task in NDN due to its intrinsic architectural characteristics. The NDN process aims to provide privacy and security to users without considering their requests and patterns. Additionally, the system encounters challenges in distinguishing between malicious and legitimate user behavior and irregular access patterns. In addition, the NDN's decentralized nature also introduces complexity because the nodes operate independently, making it challenging to identify and respond to distributed attacks. The mitigation algorithm effectively identifies cache activities; however, this reduces network efficiency and creates a computational overhead problem. The attackers mimic legitimate activity patterns and access the network to develop traffic-related issues. These difficulties result in a high delay and a low hit rate, negatively impacting the system's scalability. The research issues are addressed using the enhanced deep learning (EDL) model, which uses the deep neural network function to train the system to identify malicious user strategies and cache patterns. Additionally, the memetic optimization technique is employed to minimize the delay and deviation between the predictions. The deep neural networks in EDL are used to train the systems to identify malicious cache patterns, which increases the attack detection accuracy. During this process, the memetic optimization approach is integrated with the deep learning approach, which enhances overall reliability and performance by reducing prediction deviation and delay. The integrated CPA prediction design addresses various NDN challenges, including privacy-focused design, decentralized operations, and differentiating between legitimate and malicious access patterns.

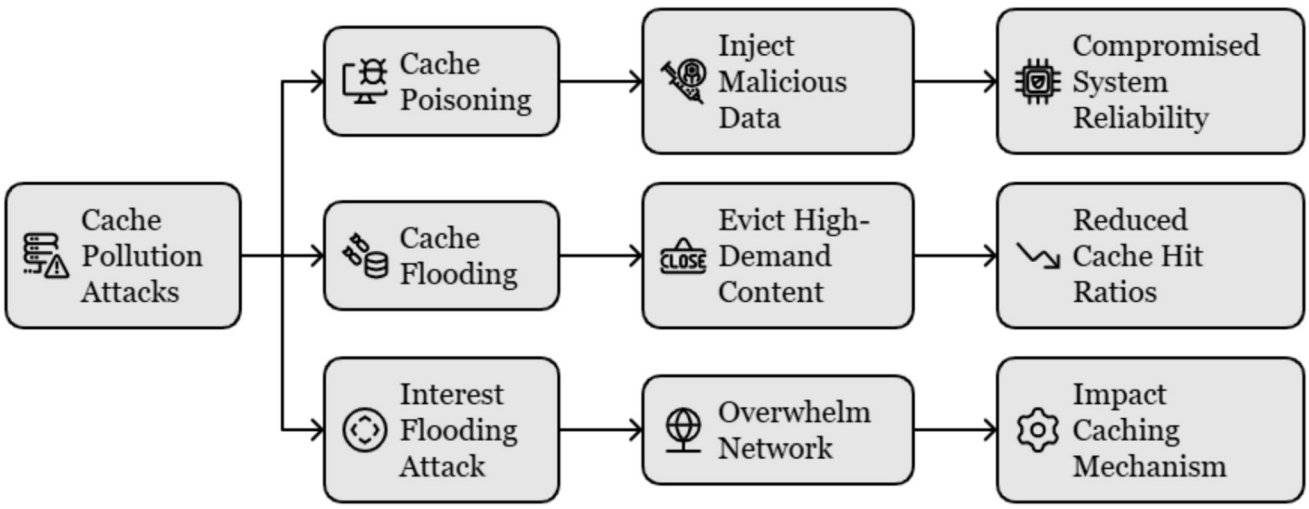

Fig. 1 Types of CPA in NDN

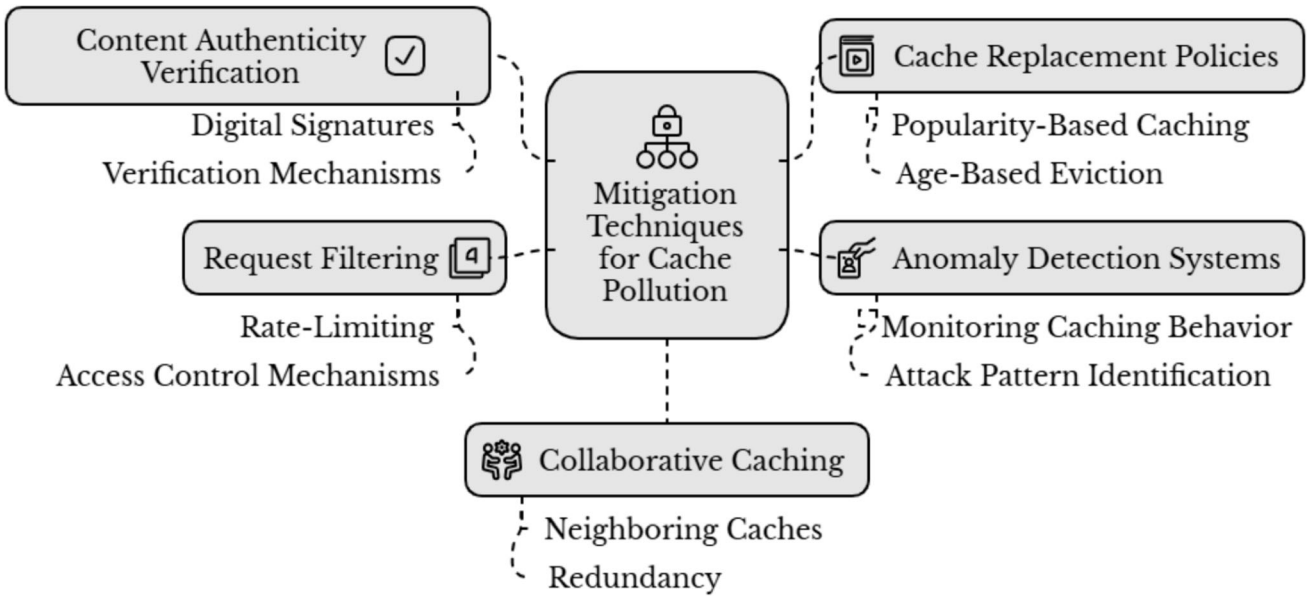

Fig. 2 CPA detection and mitigation techniques

1.2 Focus of this work

The primary focus of this work is listed below.

- To create a robust, enhanced deep learning model to categorize legitimate and malicious users' cache patterns and confirm maximum CPA prediction accuracy at dynamic network settings.
- To combine the memetic optimization technique with the DL approach to observe the cache strategies in a resource-intensive and large-scale environment for minimizing prediction deviation. This process manages scalability, reduces computational overhead, and maximizes network size.
- To develop the EDL systems by adjusting system parameters to differentiate between stealthy and coordinated behavior, which helps predict malicious activities even when using different cache policies and criteria.

The memetic optimization algorithm enhances the deep learning framework by integrating global evolutionary strategies with an adaptive local search mechanism to detect cache pollution attacks (CPAs) in dynamic named data networking (NDN) environments. Unlike conventional approaches, adaptive local search dynamically adjusts step sizes based on the magnitude of the gradient and the curvature of the loss landscape, thereby enhancing convergence stability and speed. This refinement facilitates effective exploration and exploitation, especially in large-scale, resource-intensive contexts. Convergence analysis demonstrates that the update rule satisfies the Robbins–Monro conditions under restricted gradients, thereby ensuring that the gradient norm asymptotically approaches zero. That the approach converges to a stationary point validates its accuracy and computational efficiency. The optimization algorithm also distinguishes stealthy and coordinated harmful activities under shifting popularity dynamics, exceeding existing methods.

Then, the rest of the script is systematized as follows: Sect. 2 discusses the literature exploration to understand the importance of learning techniques in predicting the CPA. Section 3 discusses the working process of the EDL-based CPA detection process and evaluates the system's efficiency in Sect. 4. Section 5 concludes the entire work.

2 Literature exploration

The present-day systematic literature review aims to analyze previously published research that identifies various types of network attacks, including those targeting wireless sensor networks, software-defined networks, and web servers. According to several studies conducted, it has been proved that the use of machine learning (ML) and deep learning (DL) techniques significantly helps in the detection of and dealing with distributed denial of service (DDoS) attacks (Tariq Emad Ali et al. [15]; Meenakshi Mittal et al., 2022 [16]). These techniques have also proven effective in identifying advanced attacks and handling large amounts of data. The papers reviewed include detailed descriptions of the existing state of detection methods, the type of datasets used, data preprocessing techniques employed, performance indicators set, and the objectives and layout of future trends in network security. Yao et al. [17] proposed an ensemble learning (EL) approach for identifying the cache pollution attacks (CPAs) in vehicle content-centric networks (VCCN). This study intends to predict the CPA by collaborating with multiple vehicles. During this process, the head vehicle consists of several parameters, such as direction, speed, and position, which help to form the clusters. The cluster head is the base learner that trains the system to identify attacks based on the hit rate. The false ratio is minimized with the help of an ensemble learning classifier, which reduces the linear optimization problem retrieval delay and improves the attack detection rate. The introduced learning approach addresses the overfitting issues when analyzing vehicle features, thereby ensuring system generalization. Wang et al. [5] recommended a request pattern change-based algorithm (RPCA) to identify the cache pollution attacks in edge computing. This study analyzes the various strategies to determine the CPA in mobile and edge devices. Initially, content requests are examined using the eavesdropping strategy. Then, the RPCA approach was applied at the request to explore patterns that effectively recognize abnormal ones. Then, an aware cache defense approach is proposed to take the appropriate actions on CPA. This procedure ensures a high hit rate and effectively minimizes deviation errors.

Hidouri [18] suggested a Q-learning approach to identify and mitigate CPA in ICAN for named data networking. The author aims to enhance the overall CPA detection rate and reduce the retrieval delay in ICAN. The objective is achieved by integrating every router with a reinforcement learning process to identify the actions in the network. The reward and penalty values help predict the hit rate and arrival time. These features identify the malicious and normal activities with 95.09% accuracy, 94% hit rate, and reduced delay (18%). Kim, H. et al. [19] applied the deep learning (DL) approach to identify the multiple cache channel attacks. This study focuses on ABORT and PRIME attacks to identify cache events. The DL approach utilizes several layers that receive network inputs and process intermediate functions, which detect integrated attacks with minimal deviation error. During the analysis, various learning classifiers, including a recurrent layer, multilayer perceptron, and long short-term memory (LSTM) neural networks, are utilized to identify cache attacks. Among the various analyses, the LSTM approach achieves the highest recognition rate for detecting the integrated cache attack. Yao et al. [20] detecting CPA using federated learning (FL) in ultra-dense networks. This study associates several small base stations to form clusters based on load and distance similarities. The cluster head trains the system to identify cluster members based on their characteristics, which impacts CPA detection efficiency. The classification process is improved by accumulating macro-stations, thereby enhancing overall clustering efficiency. In effectively utilizing the FL concept, every head works independently to manage the unbalanced issues while successfully predicting the CPA.

Kar et al. [21] created CPA detection systems using a rank comparison approach in named data networking. The anomaly or cache attack is identified by analyzing the requested packets based on the packet request rate and volume. The ranking process helps differentiate between normal and legitimate entries, achieving a maximum hit rate and minimum delay. This ranking-based detection process enhances NDA data security and signature privacy, effectively mitigating attacks. However, the system requires additional methodologies to strengthen the CPA mitigation procedure. Sameer, M. K., & Salman, M. I. (2023) [22] recommended popularity variation mechanism (PV) cache pollution detection systems to observe legitimate users. The PV system predicts cache attacks based on deviations, content popularity, and request rate. This study focuses on cache pollution, privacy,

and poisoning attack detection and mitigation processes, aiming to create robust security systems. Although the system effectively mitigates various cache attacks, it faces computational issues due to pollution attacks. Yao et al. [23] used an ensemble learning approach to detect cache pollution attacks in information-centric networking (ICN) based on vehicular ad hoc networks (VANETs). This approach attempts to combine the strengths of multiple machine learning models to enhance detection capabilities and mitigate the impact of cache pollution, which can hinder network performance and increase latency. The results of the experiment conducted revealed that the suggested method considerably increases security and reliability in VANET environments when compared to conventional detection methods.

Wang et al. [5] suggest a method for detecting and defending against cache pollution attacks in edge computing by analyzing pattern modifications. Their approach detects abnormal request content distribution patterns, considering the presence of potential attacks that may reduce the effectiveness of the cache and heighten network congestion. Those attacks increase system resource consumption and network mapping. The study confirms that the proposed approach can achieve high security for the system and manage caches efficiently, making edge computing systems less prone to attacks.

Zou et al. [24] examine the performance metrics of a neural network-based attack detector in combating possible network threats. This research explores various models and assesses their effectiveness in identifying anomalies, as well as intrusions and cache pollution, within the network. The data support the notion that streams in the form of deep learning achieve the best performance in capturing and classifying content with significant changes within the volume network, especially through the use of convolutional and recurrent networks. Kumar and Srivastava [25] introduce the interface-based popularity control (IBPC) method to address polluting attack threats within the cache in named data networking (NDN). The technique examines how changes in the request patterns of specific interfaces impact the overall merit of the content being sought, distinguishing between genuine and malicious access. Results from the experiments demonstrate that IBPC has a considerable effect on reducing the pollution of caches. According to multiple researchers, cache pollution attacks can be predicted with the help of several intelligent learning approaches. However, the system faces difficulties in processing dynamic packet requests, leading to high retrieval delays and problems in making malicious decisions. The research difficulties are addressed with the help of the enhanced deep learning (EDL) model, which identifies malicious attacks with a maximum hit rate and minimum delay. Under dynamic and unexpected packet request patterns, content-centric networking faces cache pollution attacks. Traditional intelligent learning methods, while successful, struggle to adjust in real time, which can delay retrieval and lead to misidentifying malevolent conduct. This study presents a new enhanced deep learning (EDL) model to overcome these constraints. The proposed EDL system employs a multilayer learning structure designed for real-time anomaly detection, enabling it to adapt to changing traffic patterns and differentiate between genuine and malicious requests more accurately than standard models. Adaptive feature selection and temporal behavior analysis enhance cache hit rates and reduce retrieval delays. This improvement increases cache protection techniques, network stability, and efficiency, setting a new standard for intelligent threat mitigation in content delivery systems.

3 Intelligent deep learning framework

The main objective of this work is to detect the cache pollution attacks (CPAs) in NDN using the enhanced deep learning (EDL) framework. The EDL framework integrates the memetic optimization algorithm with a deep learning model to identify malicious requests in the networks. The malicious requests cause performance degradation and flooding issues, affecting data security and privacy. The system objective is achieved by exploring network features such as temporal variations, cache hit ratio, and request patterns, which help identify the CPA with the maximum detection rate. During the analysis, a deep learning model was utilized to train the system, reducing the error rate and addressing optimization problems. The optimization process reduces the retrieval delay (r_d) and false positive rate with the help of a memetic optimization approach that fine-tunes the

network parameters according to the local and global searching strategies. The parameter updating procedure ensures the system's robustness, flexibility, and scalability while detecting CPA attacks. The overall structure of the CPA prediction process is illustrated in Fig. 3.

Figure 3 illustrates the architecture representation of the EDL-based CPA detection process in the NDN environment. The system receives the input from the NDN traffic sources because various users (malicious m_u and legitimate l_u) send the request data to the router (R_n) to access the information from the processor. The received data packets (P_a) is processed by several layers to predict the m_u and l_u by exploring the CPA in NDN. The P_a is normalized to get the various features, such as access frequency $f(c)$, content diversity $D(t)$, temporal pattern (T_i), cache hit rate (C_{hr}), and anomaly indicator (a_i). These features are examined by recurrent networks that effectively process the sequence of P_a and identifies the m_u and l_u with maximum prediction rate. During the analysis, a memetic optimization algorithm is incorporated to perform the local l_s and global g_s search to choose the optimized parameter used to minimize the output deviations (o_d). Therefore, the main goal of this work is to analyze the NDN traffic (NDN_{tX}) and classifier into normal traffic (Tx_{lu}) and malicious (Tx_{mu}) with minimum cost (c) and maximum accuracy (a). Let us consider the traffic information $tX = \{x_1, x_2, \ldots x_n\}; x_i \in \mathbb{R}^d$. Here, x_i is represented as features which is derived from P_a and the EDL classification is defined as

$$f_\Theta(x_i) = \begin{cases} Tx_{mu} & otherwise \\ Tx_{lu} & if\ P(Tx_{lu}|x_i) > P(Tx_{lu}|x_i) \end{cases}$$

The overall objective of this work is then defined in Eq. (1).

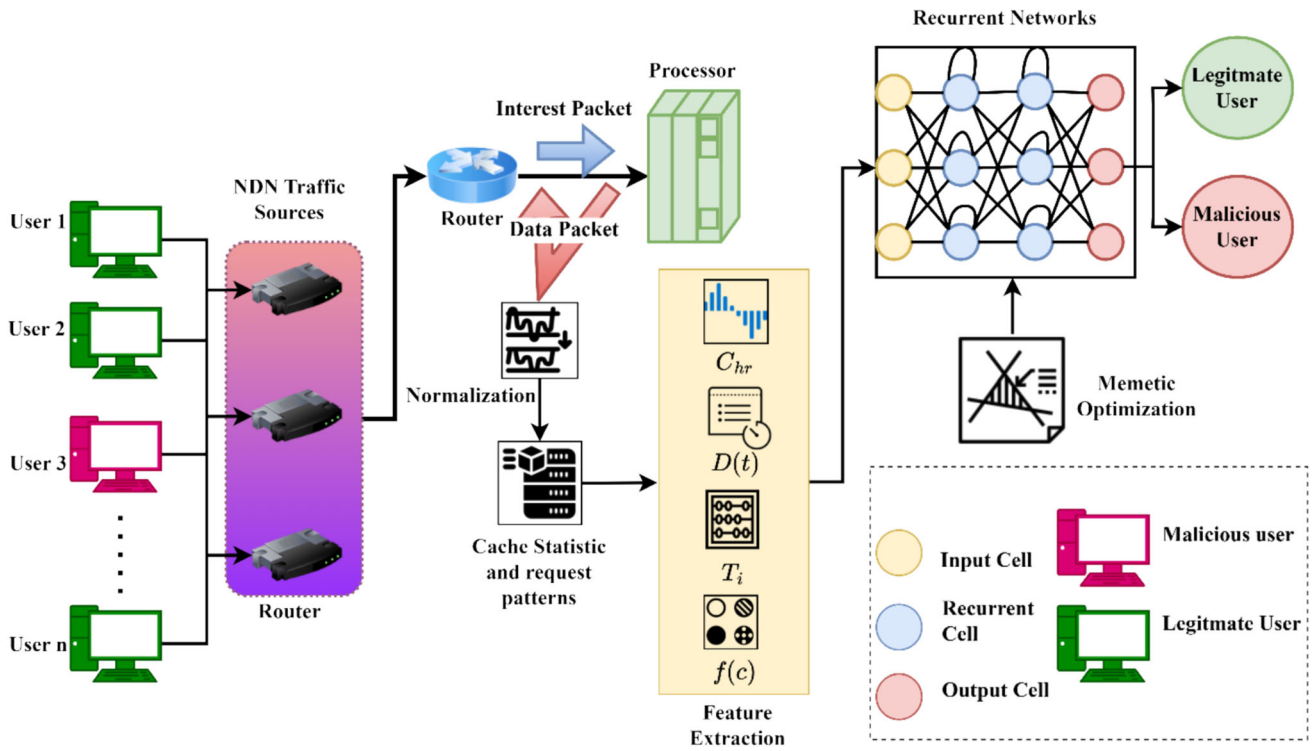

Fig. 3 Architecture of EDL-based CPA detection

$$\left.\begin{array}{c} \max_{\Theta} a - \lambda_1 c - \lambda_2 \mathcal{L}_p \\ a = \dfrac{tp + tn}{\text{total instance}} \\ c = g(\Theta) \\ \mathcal{L}_p = \alpha.fp + \beta fn \\ \Theta^* = \arg \min_{\Theta} \mathcal{L}_{\text{total}} \\ \mathcal{L}_{\text{total}} = \mathcal{L}_{cl} + \mathcal{L}_p + \mu.\mathcal{L}_{\text{reg}} \end{array}\right\} \tag{1}$$

According to Eq. (1), objective $max(a)$ is achieved by computing the true positive (tp), true negative (tn), and total instances. The a computation considers the $\mathcal{L}_p$ misclassification penalty and hyperparameters λ_1 *and* λ_2 to ensure the minimum computational cost $c = g(\Theta)$. The Θ is improved with the help of the memetic optimization that utilizes the regularization parameter μ for reducing the overfitting and computational issues in NDN.

3.1 P_a Preprocessing block

The objective of this block is to clean P_a in NDN traffic data and preparing features for identifying m_u. The P_a data is defined as $\{x_1, x_2, \ldots . x_n\}$, for every x_i consists of important information like content name (c), time stamp t_i, packet type (p_t), source id (s_i) and response status R_s. The gathered P_a is examined to eliminate the irrelevant information at a small time window (Δt) using the below condition and the filtered P_a is defined as in Eq. (2)

Conditions:

$$\textbf{\textit{Cond}}\quad 1 : \textit{If}\quad x_i \quad \& \quad x_j \quad \textit{satisfy} \quad c_i \quad = c_j \quad \& \quad \left|t_i - t_j\right| \leq \Delta t \quad \textit{then} \quad \textit{remove} \quad x_j$$

$$\textbf{\textit{Cond}}\quad 2 : \textbf{\textit{if}} \quad c \notin re_{\text{cont}} \; \textit{set}\; R, \quad \textit{then} \quad \textit{removed}\; x_i$$

$$\textbf{\textit{Cond}}\quad 3 : \textit{if} \quad x_i \neq p_a \quad \textit{standard} \quad \textit{then} \quad \textit{remove} x_i$$

$$\textbf{\textit{Cond}}\quad 4 : \textit{if} \quad t_i < T_{start} \quad \textit{or} \quad t_i \quad T_{end}\,, \textit{then} \quad \textit{remove} \quad x_i$$

According to the ***cond*** 1 the duplicate c is removed from $X \in p_a$ is removed as $X' = X \backslash \{x_j : c_i = c_j \, and \, |t_i - t_j| \leq \Delta t\}$. Then ***cond*** 2 is applied to remove the irrelevant data to maintain the target *cinR* and the filtered data is $X'' = X' \backslash \{x_i : c \notin R\}$. Afterward ***cond 3*** is utilized for filtering the anomalous information or incomplete data. The condition checks the $X \in p_a$ in terms of valid packet names, (s_i), (p_t), and (t_i). The ***cond*** 3 produces the filter data as $X''' = X'' \backslash \{x_i : malformed(x_i) = True\}$. Finally, ***cond*** 4 is applied to identify the traffic window $[T_{start}, T_{end}]$. The filtering data is defined as $x_{final} = X''' \backslash \{x_i : t_i < T_{start} or t_i > T_{end}\}$. Then, the combined filtering process is defined in Eq. (2)

$$\begin{aligned} X_{fi\,data} = X \backslash \{x_j : c_i = c_j and \left|t_i - t_j\right| \leq \Delta t\} \backslash \backslash \quad \{x_i : c \notin R\} \backslash \{x_i : malformed(x_i) = True\} \\ \{x_i : t_i < T_{start} \quad or \quad t_i > T_{end}\} \end{aligned} \tag{2}$$

The $X_{fi_{data}}$ data is analyzed using a normalization process that converts the X data into a unified format, which is computed as $X' = \frac{x_i - \mu}{\sigma}$. After normalizing the X, features like access frequency $f(c)$, content diversity $D(t)$, temporal pattern (T_i), cache hit rate (C_{hr}), and anomaly indicator (a_i). The below conditions are utilized to derive the feature from $X \in p_a$.

Conditions:

$$\textbf{\textit{Cond}} \quad 1 : c \in C_{valid} // f(c)$$

$$\textbf{\textit{Cond}} \quad 2 : t_i \in [t_1, t_2] // D(t)$$

$$\textbf{\textit{Cond}} \quad 3 : \boldsymbol{c_i} = \boldsymbol{c_i} + \boldsymbol{1} // \boldsymbol{T_i}$$

$$\textbf{\textit{Cond}} \quad 4 : \boldsymbol{C_r}(\boldsymbol{x_i}) \in \{\boldsymbol{Hit}, \boldsymbol{Miss}\} // \boldsymbol{C_{hr}}$$

$$\textbf{\textit{Cond}} \quad 5 : z - score(f(c)) > \tau // (a_i)$$

Based on the above *cond*1, the $f(c)$ is derived, which extracts the number of requests for a particular c, which is obtained as $f(c) = \sum_{i=1}^{n} 1(x_i = c)$. In the $f(c)$ computation, $1(x_i = c)$ verifies that x_i the packet is relevant to the requested content c. Then, *cond*2 is utilized to obtain the c variety at time$t$. The $D(t)$ is obtained from the content name of $p(a)$; $D(t) = |\{c_i : x_i \in X' at\ t\}|$. The *cond*3 is utilized while extracting T_i for interest packets $T_i = t_{i+1} - t_i$. This T_i feature used to identify malicious traffic because it has small intervals. *Cond*4 *and* 5 used to obtain the C_{hr} and a_i. If c is presented in the cache, which is c_{hr} else miss and the c_{hr} is estimated as $\frac{number\ of\ cache\ hit}{total\ request}$. Then, the cache irregularities are identified with the help of *cond*5, which is estimated with the help of

$$\begin{cases} 1 & if \quad z - score(f(c)) > \tau \\ & 0 \quad otherwise \end{cases}; \quad z - score = \frac{x - \mu}{\sigma}$$

. Finally, the overall feature representation is defined in Eq. (3a)

$$\Phi(x_i') = \left[f(c), D(t), T_i, C_{hr,a_i}\right] \tag{3a}$$

$$f_\Theta(\Phi(x_i')) = \begin{cases} Tx_{mu} & otherwise \\ Tx_{lu} & if \quad P(Tx_{lu}|\Phi(x_i')) > P(Tx_{mu}|\Phi(x_i')) \end{cases} \tag{3b}$$

The extracted $\Phi(x_i')$ used to achieve the overall objective of the work, which is defined in Eq. (3b). Then, the graphical analysis of p_a in the preprocessing block is shown in Fig. 4.

Figure 4 illustrates that the graphical representation of P_a in the preprocessing block. The analysis uses the access frequency distribution (f_{req}) which is explored with the content name (c_n). According to the c_n, $D(t)$ is examined for every request with t_i. In addition, the T_i is examined for every c_n at which some intervals do not receive any packets that indicate no traffic events. In addition, the preprocessed data was explored to determine the hits c_h and misses c_m which are utilized to analyze the Tx_{mu} *and* Tx_{lu} in the NDN traffic. The derived features fed into the classifier to categorize the Tx_{mu} *and* Tx_{lu}. To ensure the dataset is of sufficient quality, trustworthiness, and uniformity for training the deep learning detection model, a multi-phase preprocessing pipeline was employed. Data filtering was implemented according to four primary conditions (Cond 1–4): (1) removal of incomplete records containing missing or null values; (2) elimination of duplicate entries to decrease noise; (3) selection of traffic pertinent to cache interaction based on named data networking metadata; and (4) outlier exclusion based on the interquartile range (IQR) method. To ensure balanced feature influence during training, continuous numerical features were scaled using Min–Max normalization to [0,1] for normalization, while categorical features such as protocol types and attack labels were one-hot encoded to maintain their categorical nature and avoid ordinal bias. For high-dimensional sparse data, such as request content types, neural networks with embedding layers were applied to capture latent patterns and reduce dimensionality. Using stratified sampling, the dataset was divided into training (70%), validation (15%), and test (15%) sets, while preserving the class

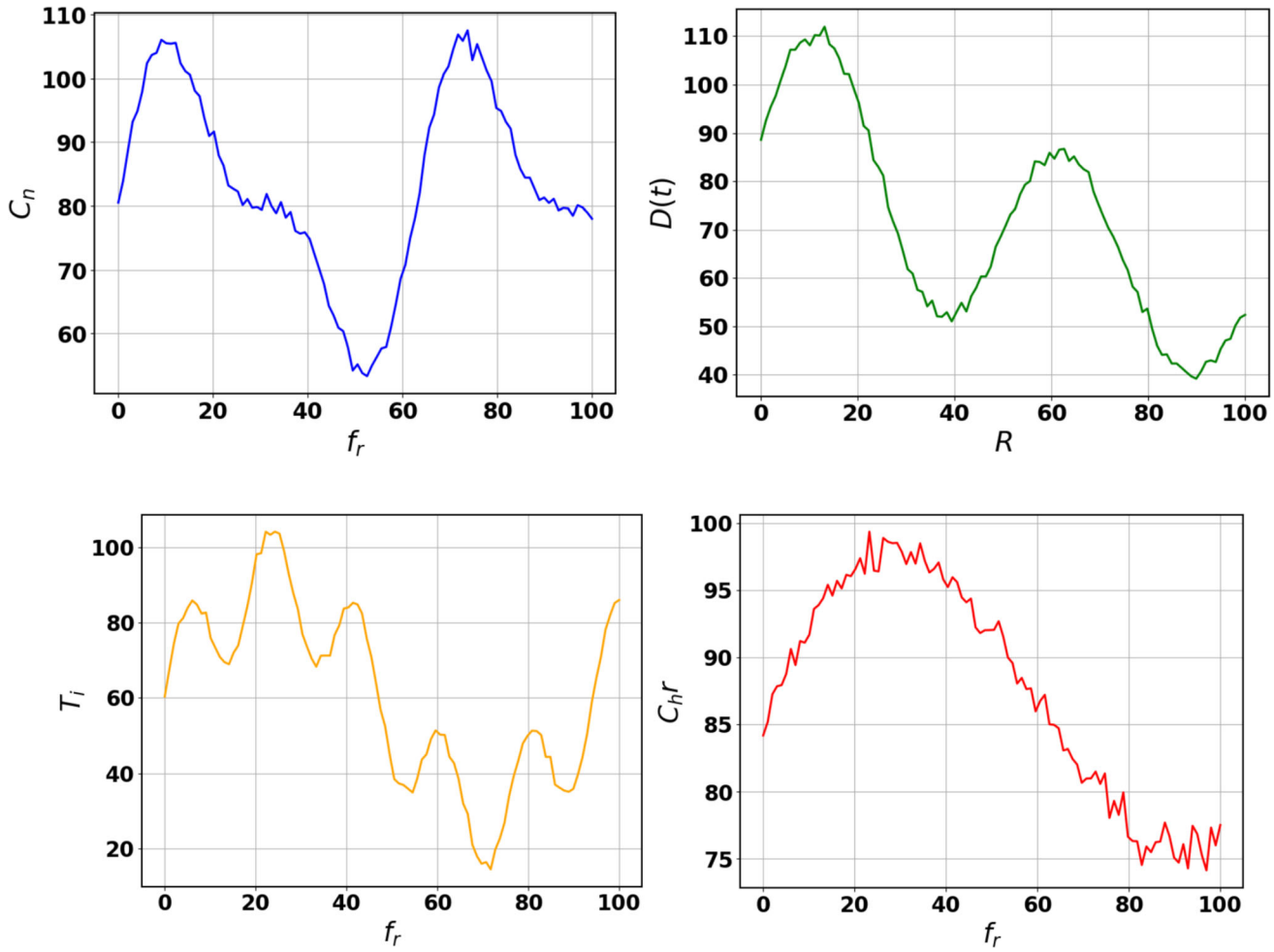

Fig. 4 Visualization analysis of $\boldsymbol{P_a}$

distribution. Furthermore, a time-series-aware split was performed to maintain temporal causality, simulating real-world scenarios. This thorough preprocessing approach ensured integrity while enabling efficient learning and generalization of the proposed detection framework. The detailed working process is described below.

3.2 EDL process for CPA classification

The next important phase is CPA classification, which applies the enhanced deep learning (EDL) approach. The EDL approach uses the recurrent network with a memetic optimization algorithm to classify the $\Phi(x_i') = [f(c), D(t), T_i, C_{hr,} \text{ and } a_i]$ into Tx_{mu} *and* Tx_{lu}. Then, the process of recurrent networks is shown in Fig. 5.

The generated $\Phi(x_i')$ are unified by applying the $norm(\Phi(x_i'))$ which predicts the $\min(x_i)$ and $\max(x_i)$. Then the $norm(\Phi(x_i'))$ is computed as $\frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$ which is denoted as x_i^{norm}. The x_i^{norm} value is fed into the recurrent layer to identify the temporal dependencies, and the parameter utilized in the classification process is defined in Table 1.

The received input x_i^{norm} is further defined as $X = [x_1^{\text{norm}}, x_2^{\text{norm}}, x_3^{\text{norm}}, x_4^{\text{norm}}, x_5^{\text{norm}}]$ is fed into the 1st hidden state h_t which uses the W *and* b and produces the $Z^{(1)}$ as output by utilizing the ϕ. The $Z^{(1)}$ is fed into the next state h_{t+1} to compute the $Z^{(2)}$. From the computed $Z^{(2)}$ value final output is predicted, and the computation of h_t is defined in Eq. (4)

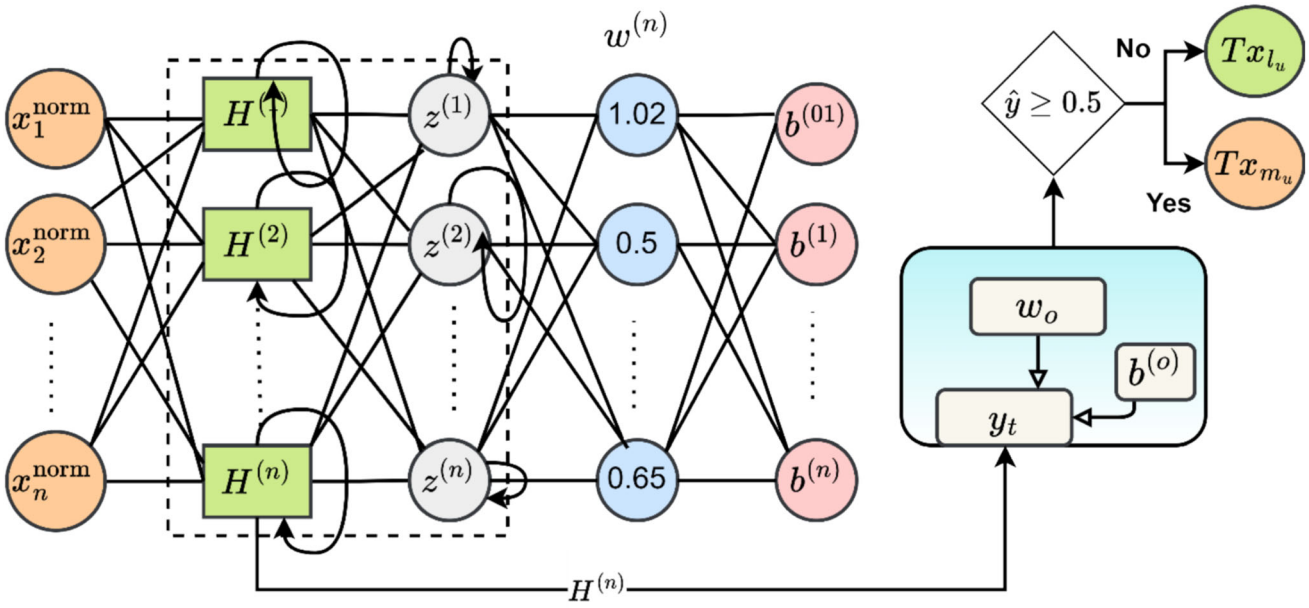

Fig. 5 Recurrent neural network structure

Table 1 Parameter description

Parameter	Description
x_i^{norm}	Normalized input
h_{t-1}	A previously hidden layer at t time stamp
h_t	t time stamp hidden layer
W_h	Hidden state recurrent weight value
$f(c)_t$	t time stamp access frequency, which is weighted by $W_{f(c)}$
T_i	Request's temporal pattern by W_{T_i}
$D(t)_t$	t time stamp content diversity weighted by $W_{D(t)}$
C_{hr}	Cache hit rate by $W_{C_{hr}}$
ϕ	ReLu activation function
a_i	Anomaly indicator by W_{a_i}
b_h	Hidden state bias value
W^1	Dense layer one weight value
b^1	Dense layer one bias
W^2 and b^2	Dense layer two weight and bias
$Z^{(1)}$ and $Z^{(2)}$	Dense layer 1 and 2 pre-activation output

$$\left.\begin{aligned} H^{(1)} &= \phi\left(Z^{(1)}\right) = \phi\left(W^1.X + b^1\right) \\ H^{(2)} &= \phi\left(Z^{(2)}\right) = \phi\left(W^2.H^{(1)} + b^2\right) \\ Z^{(o)} &= W^o.H^{(2)} + b^o \end{aligned}\right\} \tag{4}$$

The $X = \left[f(c), D(t), T_i, C_{hr}, a_i\right]$ is processed in the recurrent layer, and the obtained h_t using Eq. (5) and final output $\widehat{y}$ is computed using Eq. (6). The computed output h_t is fed into the output layer $\widehat{y}$ that uses the ReLU activation function to predict the NDN traffic status.

$$\left.\begin{array}{c} h_t = \phi\left(W_h.h_{t-1} + h_{f(c)} + h_{D(t)} + h_{T_i} + h_{a_i} + b_h\right) \\ h_{f(c)} = W_{f(c)}.f(c)_t \\ h_{D(t)} = W_{D(t)}.D(t)_t \\ h_{T_i} = W_{T_i}.T_i \\ h_{C_{hr}} = W_{C_{hr}}.C_{hr} \\ h_{ai} = W_{ai}.a_i \end{array}\right\} \tag{5}$$

$$\widehat{y} = \begin{cases} \sigma\left(W^{(o)}.\phi\left(W^{(2)}.\phi\left(W^{(1)}.\left[f(c), D(t), T_i, C_{hr,} a_i\right] + b^{(1)}\right) + b^{(2)}\right) + b^{(o)}\right) \\ \sigma(w_o.h_t + b_o) \end{cases} \tag{6}$$

According to Eq. (6), the predicted $\widehat{y}$ value is compared with the threshold value to detect the Tx_{mu} *and* Tx_{lu}. The detection is done as $Cl = \begin{cases} Tx_{mu} & if\, \widehat{y} \geq 0.5 \\ Tx_{lu} if & \widehat{y} < 0.5 \end{cases}$; this Cl process's main intention is to reduce the deviations $(\mathcal{L})$. The $\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\widehat{y_i}) + (1 - y_i)\log(1 - \widehat{y_i}]$. If the system receives a high $\mathcal{L}$ value, then the network parameters W *and* b should be updated via the local l_s and global g_s search. The searching process improves the neural training process and improves the overall (a). During the analysis, g_s uses the genetic operators to identify the near-optimal solutions and l_s uses the gradient techniques to update the parameter recognized by g_s. The search process enhances overall data analysis and mitigates high computational problems and overfitting issues. The first step of this work is candidate solution initialization $CS = \{p_1, p_2, p_3, \ldots p_N\}$; N is population size. Every p_i denoted that the recurrent network parameters $p_i = \left[W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, W^{(3)}, b^{(3)}, \ldots W^{(o)}, b^{(0)}\right]$. After that, fitness function $F(p_i)$ is computed for p_i to identify the optimized parameters for minimizing the $\mathcal{L}(\widehat{y_i}, y_i)$. Therefore, g_s is performed by applying crossover and mutation operators to get the new solution by analyzing parent solutions (p_i, p_j). After that, l_s is performed in which gradient refinement is done to get the optimal solution. The selected parameters are arranged in the search space and compared with $F(p_i)$ for getting the best solution. The effective utilization of l_s *and* g_s maintains the stability between exploitation and exploration. This searching process is defined in Eq. (7)

$$\left.\begin{array}{rl} Po^{t+1} = select((g_s(Po^t) \cup l_s(PO^t)) - position & update \\ p_c^{t+1} = \alpha.p_i^t + (1 - \alpha).p_j^t + \delta, \delta \sim N(0, \sigma^2) - g_s & process \\ p_i^{t+1} = p_i^t - \eta.\nabla L\left(p_i^t\right) - l_s & process \\ F(p_i) = -\dfrac{1}{N}\displaystyle\sum_{j=1}^{n} \mathcal{L}(\widehat{y_i}, y_i) - fitness & fucntion \end{array}\right\} \tag{7}$$

According to the above process, the selected W *and* b parameters are updated frequently to reduce the $\mathcal{L}(\widehat{y_i}, y_i)$. The effective utilization of l_s *and* g_s minimize the local minima and improve the overall CPA detection a. In addition, the integrated analysis ensures robustness and maximum convergence rate while predicting cache pollution attacks. According to this process, the pseudocode of the CPA detection process is shown in Table 2.

According to the algorithm steps, the optimized parameters are selected from the search space by combining the g_s *and* l_s to improve the overall network output $\widehat{\mathrm{y}}$. The NDN network analyzes cache activity data, including cache hit rates, miss rates, requests, and access patterns. This data is explored to extract the features $\left[\mathrm{f(c)},\ \mathrm{D(t)}, \mathrm{T_i}, \mathrm{C_{hr,}}, \mathrm{a_i}\right]$ to identify the pattern for predicting the CPA malicious activities. Then, features are normalized to ensure the input is consistent during data training. For every t time stamp, hidden states computes the output $\widehat{\mathrm{y}}$ and the deviation between the outputs are reduced by estimating the fitness function $F(p_i) = -L(\widehat{y_i}, y_i)$ according to the local and global search. The selected parameters $\mathrm{Op_{para}}$ help to minimize the overfitting issues, and intricate temporal dependencies and convergence while updating the network parameters. Therefore, the system recognizes the anomalies and attacks with maximum accuracy and ensures robustness

while exploring NDN_{tX}. In addition, the Op_{para} in neural networks improves the cache demands, which leads to maximize the cache hit ratio C_{hr} and minimize the retrieval delay (r_d). Then the C_{hr} for various numbers of epochs and configurations is analyzed, and the results obtained are shown in Fig. 6.

Figure 6 illustrates that the C_{hr} analysis while updating the network parameters using the memetic optimization algorithm Op_{para}. The selected Op_{para} utilized for fine-tuning the parameters such as *learning rate*(ρ), *number of neurons*(n), regularization parameter (δ), weight (w_{ij}), and bias (b_i). These parameters are fine-tuned continuously to enhance the classification a_i while observing the C_{hr} during the training process. The analysis explores the frequency distribution of C_{hr} over the epochs. Configurations of 1 to 20 parameters for every analysis are fine-tuned based on optimization algorithms. The explored histogram was used to analyze the spread (range of C_{hr}), peaks (frequency of C_{hr}), and skewness (distribution status). The distribution of the C_{hr} is used to identify the low and high performance of the classifiers while identifying the CPA attacks. In configuration 1, the model attains a narrow peak with a high C_{hr} which indicates that the model ensures efficient and stable cache management. In configuration 5, the distribution of C_{hr} is mid-range (0.4–0.6), which shows inconsistent performance because of the suboptimal parameter updating process. Then, in configuration 10, the model ensures the lower C_{hr} which indicates the model is trying to adapt the network parameters. Thus, the selected Op_{para} helps to manage the classifier performance, which is clearly shown in the C_{hr} analysis. The EDL method not only ensures the high C_{hr} it also attains the minimum retrieval data while requesting the cache in the NDN network environment. Then the obtained r_d is shown in Fig. 7.

Figure 7 shows r_d analysis of the EDL approach concerning various epochs training in the CPA detection process. The selected Op_{para} creates an impact on r_d on the cache management process. From the analysis, configurations 1–5, the method attains r_d minimum value and low variability. The minimum r_d indicates that the model utilizes the optimized parameters, ensuring robustness while managing the cache. The configuration moved into a higher range from 15 to 20; then, the r_d attains fluctuations while training the data in the NDN network, sometimes leading to stability issues. However, frequent training minimizes the delay and improves attack detection efficiency. In addition, memetic parameters and training attributes are utilized to enhance the overall CPA attack detection process. The attained results are illustrated in Fig. 8.

Figure 8 shows that the memetic optimization impact on C_{hr} while predicting the cache pollution attack. The efficiency of the method is evaluated over different parameters such as population size (I_{ps}), mutation rate (m_r), and crossover rate (c_r) impact on C_{hr}. These parameters are utilized to fine-tune the network parameters, which helps manage the cache according to user requests. The graph 8 indicates that the C_{hr} is a relative constant for each epoch as it stays near 95.6%. Such moderate performance is only expected as the default parameters are designed to strike a balance between exploration efforts and exploitation. The baseline hit rate seems to follow a curve, which is slightly decreasing. This is possibly evidence that the system was defected gradually or that the metrics used to measure the system had improved over the same period. On the other hand, when I_{ps} the C_{hr} continues to go up. The graph also indicates that the scatter points experienced smaller fluctuations when the population size was more significant than the preceding parameters, which led to more and better solutions being available. It can be inferred that the I_{ps} the better the performance will be after the evolutionary algorithm has completed searching for solutions. Furthermore, the baseline hit rate remained constant, indicating that the increase in population size does not incur additional overhead for the system. These findings demonstrate the intricate arrangement that should be considered when configuring the evolutionary algorithms for cache optimization. The performance improves substantially when the population size is increased; however, higher mutation rates, although necessary, can lead to instability. A combination of I_{ps} and C_{hr} mutations appear to optimize trade-offs by improving performance levels with some fluctuations. At the same time, configurations with smaller populations and low m_r do not perform well in optimal search and have a slow convergence rate, thus indicating the effect of tuning such parameters on cache effectiveness. Therefore, the frequent training of parameters using Op_{para} improves the overall cache management according to the user's request.

Table 2 Algorithm for CPA detection process

Algorithm for Recurrent Networks

Input: $X=[f(c), D(t), T_i, C_{hr}, a_i]$

Output: $\hat{y}$

Initialize: w_h, w_x, b_h, h_θ

For each **t in** seq
- **Compute** h_t
  - $h_t = \phi(w_h . h_{t-1} + w_x . X_t + b_h$
- **Store** h_t

Pass final h_t to d_l
- $\hat{y} = w_{out} . h_t + b_{out}$

Apply $\phi(f(x))$

Return $\hat{y}$

Algorithm for Memetic Optimization

Input: $CS = \{p_1, p_2, p_3, \ldots . p_N\}$, G_m, $F(p_i)$

Output: Op_{para}

Initialization: CS

Evaluate $F(p_i)$ ***of*** $init_{CS}$

For each p_i ***in CS***
- ***Compute*** $F(p_i) = -\mathcal{L}(\hat{y}_i, y_i)$

Repeat until to reach G_m

Exploration
- ***For selected pa of*** (p_i, p_j)
  - $\alpha . p_i^t + (1-\alpha) . p_j^t$ *// crossover*
- ***For each*** p_i ***in offspring***
  - $p_{mut} = p_i + \delta$

Exploitation
- ***For each*** p_i
  - ***Update using gradient***
    - $p_i^{new} = p_i - \eta * \nabla F(p_i)$

Combine updated sol to form new CS

Select top N sol using $F(p_i)$

Identify b_{sol}
- $p_{best} = argmax\,(F(p_i) \forall\, p_i\ in\ CS)$

Return p_{best}

Algorithm for Enhanced Deep Learning model-based CPA

Input: $X=[f(c), D(t), T_i, C_{hr}, a_i]$, w_h, w_x, b_h, h_θ $CS = \{p_1, p_2, p_3, \ldots . p_N\}$, G_m, $F(p_i)$

Output: $\hat{y}$ *and* OP_{para}

Step 1: → **Collect** $NDN_{tX}(P_a)$

Step 2: → **extract** $f: [f(c), D(t), T_i, C_{hr}, a_i]$

Step 3: → $norm(f)$

Step 4:→**Recurrent Networks**

For each **t in** seq
- **Compute and Store** h_t

Pass final h_t to d_l
- **Compute** d_{l1}, d_{l2} ***and*** $\hat{y}$

Compute $\mathcal{L}(\hat{y}_i, y_i)$

Step 5:→**Memetic optimization**

Evaluate $F(p_i) = -\mathcal{L}(\hat{y}_i, y_i)$

Repeat until to reach G_m

Exploration
- ***For selected pa of*** $(p_i, p_j) \rightarrow \alpha . p_i^t + (1-\alpha) . p_j^t$
- ***For each*** p_i ***in offspring***→ $p_{mut} = p_i + \delta$

Exploitation
- ***For each*** $p_i \rightarrow p_i^{new} = p_i - \eta * \nabla F(p_i)$

Combine and select $b_{sol} \rightarrow p_{best} = argmax\,(F(p_i) \forall\, p_i\ in\ CS)$

Step 5→ ***classification***

Update RNN with Op_{para}

Return $\hat{y}$

Fig. 6 Analysis of C_{hr}

4 Findings and analysis

This section analyzes the efficiency of the EDL-based collusion pollution attack detection process. The ndnSIM simulation framework and hierarchical topology are utilized to develop real-time applications during the analysis. The contents are distributed in uniform and skewed patterns in which both access the traffic patterns m_u *and* l_u. The m_u create the attack in the form of interest flooding, access frequency manipulation, and content poisoning, which makes cache pollution attacks. The CPA is identified by extracting the features $f(c), D(t), T_i, C_{\text{hr}}$, *and* a_i which are fed into the recurrent networks that identify the temporal patterns by analyzing the p_a and interest data. The network performance is optimized with the help of a memetic optimization algorithm that selects the Op_{para} to fine-tune the network parameters. For every run, 10–30 min is taken to complete the cache management process. The duration (s) is determined depending on the traffic complexity and network size, and the attack injection rate is changed from 10% *to* 20% *and* 50% *to evaluate the system's* efficiency at various conditions. The efficiency of the developed system is explored using different metrics, such as cache hit rate (C_{hr}), cache efficiency (C_{eff}), false positive rate (E_{fpr}), false negative rate (E_{fnr}), error rate (err), and retrieval rate (r_d). These simulation settings are developed using the Python tool that supports the PyTorch libraries and an Intel Core i7/i9 processor, 32 GB RAM, and an NVIDIA RTX 2080/3090 GPU hardware configuration is used. The network uses 64 batch size layers, a learning rate of 0.001, 100 epochs, and 2 dense layers with 128 hidden units to obtain

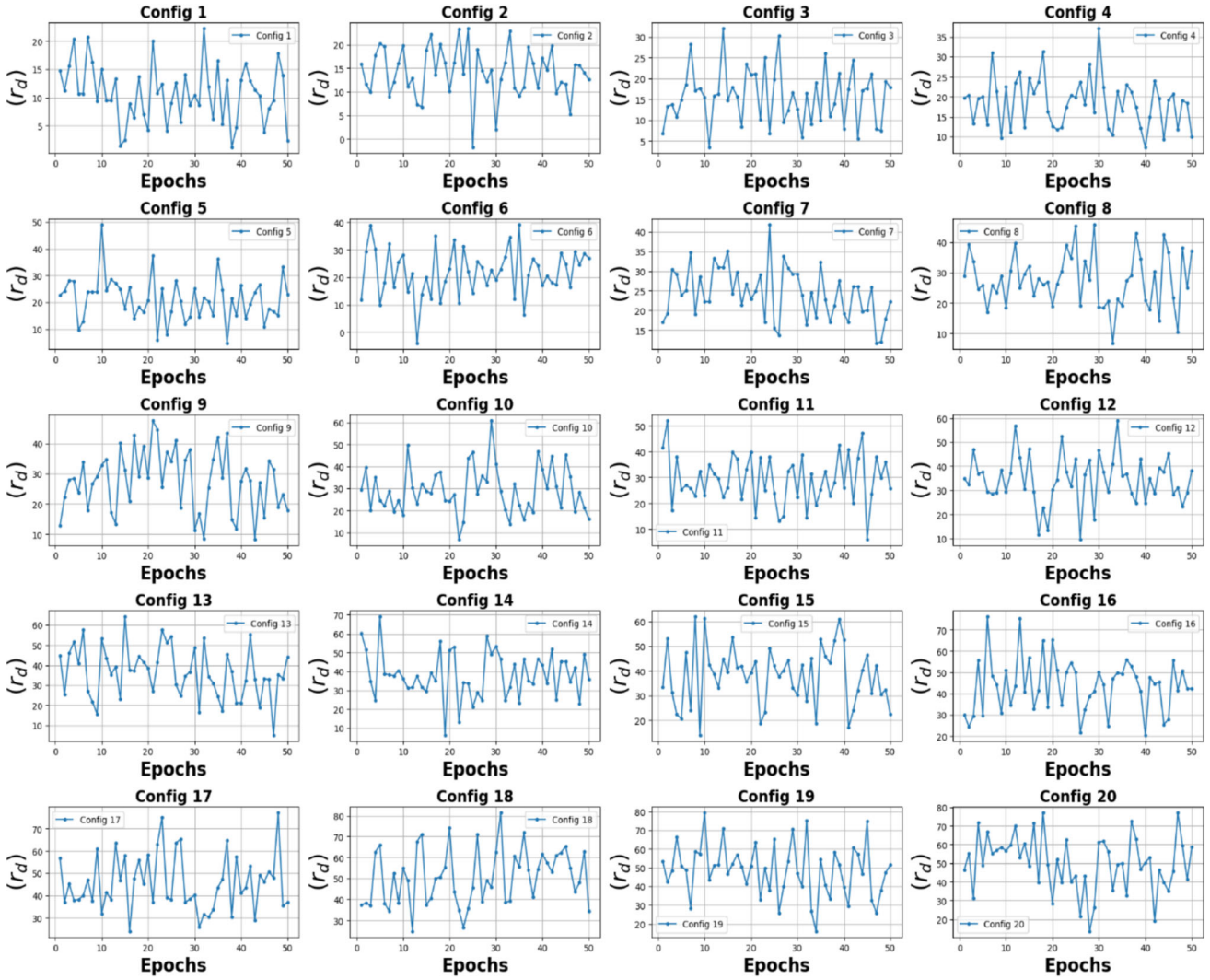

Fig. 7 r_d analysis

the output. The optimization technique uses a population size of 20, 50 generations, a crossover rate of 0.8, and a mutation rate of 0.2 for implementation. This study uses the CICIDS 2017 Dataset (https://www.kaggle.com/datasets/sateeshkumar6289/cicids-2017-dataset) [26] to evaluate EDL efficiency. The dataset consists of parameters such as flag counts, interval time, packet length, flow duration and network behavior. The dataset also includes active–idle times, flow indicators, and statistical measures that help identify malicious users when sending requests in the network environment. The dataset comprises 2.8 million records collected from July 2017, covering various attacks such as flooding, web attacks, and infiltration. Every record has 79 features that help identify malicious and legitimate users. Then, the visualization representation of the dataset is shown in Fig. 9.

The gathered data is processed using the EDL traffic preprocessing blocks, which eliminate noise and redundant information to reduce overfitting issues while examining the CPA in the NDN environment. The efficiency of the EDL is compared with different algorithms described in Sect. 2: ensemble learning (EL) [17], request pattern change algorithm (RPCA) [5], and Q-learning ICAN (QICAN) [18]. According to the analysis, the flow duration distribution has a mean value of 686 ms, with a standard deviation of 972.51. The method ensures that the information flow occurs effectively in both the forward and backward directions, at 20 packets and 15 packets, respectively. Then, the packet length distribution pattern has a mean value of 1781.53 bytes for forward packet length and 1090.57 bytes in a right-skewed distribution. The system ensures that the maximum packet

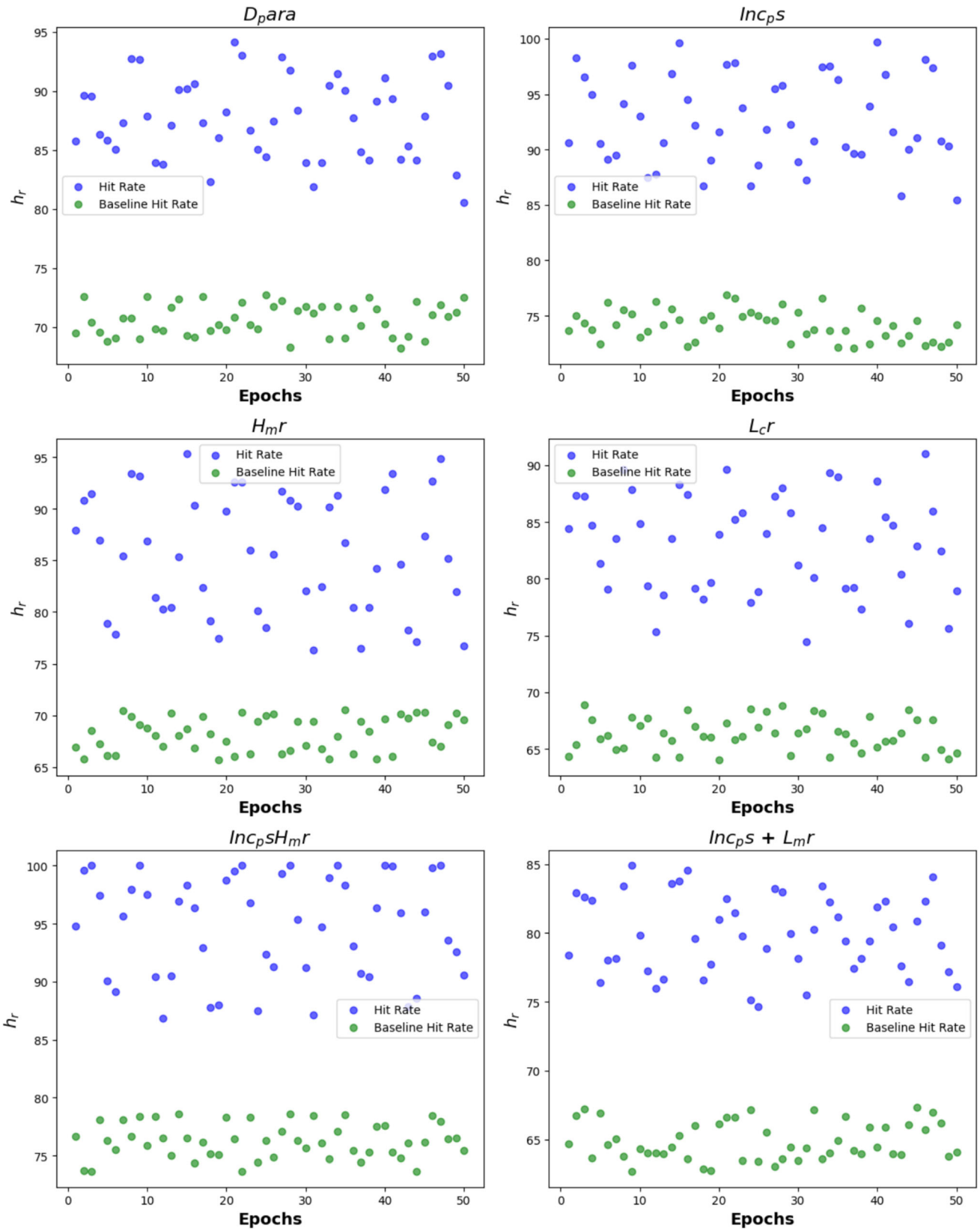

Fig. 8 Memetic optimization impact on C_{hr}

sizes are consistent, which is a forward packet length maximum distribution of 397 bytes. Then, the model extends the IAT max distribution rate to 7000 ms, indicating that the packet has a specific flow while distributing data in the network environment. The result obtained from the implementation setup is shown in Table 3. The analysis is performed for different attack injection rates (10%, 20%, 30%, 40% and 50%).

Table 3 illustrates the efficiency analysis of EDL while predicting the CPA in the NDN environment. The EDL efficiency is examined in various var_{rate} and n_s in which the EDL method achieves high C_{hr} achieve rates of up to 99% compared to other methods. In addition, if the user enters into the system method to recognize the malicious users m_u and legitimate user l_u with a maximum recognition accuracy of 98% above. The effective parameter utilization procedure reduces the retrieval delay and deviation between the outputs. The enhanced deep learning (EDL) stands out due to its unique memory organization technique, which efficiently utilizes intelligent caching mechanisms. The EDL achieves 96–99% hit rates, significantly higher than the 75–80% averages of legacy systems. In addition, EDL has a two-level cache structure, whereby the L1 cache is used for data that are used more often, allowing for fast access. At the same time, the L2 comprises data about a more extended range dependency, which allows limited access. This interplay between predictive caching and a multi-tiered memory architecture enables EDL to perform well on many workloads. Then, the efficiency of the EDL approach is compared with other existing methods, such as federated learning (FL) [20], the popularity variation mechanism (PV) [22], and the interface-based popularity control (IBPC) [25]. The obtained results are then shown in Fig. 10.

The proposed EDL (efficient deep learning-based caching system) outperforms FL, PV, and IBPC in all primary metrics, confirming its suitability for intelligent caching systems. EDL maintains highly relevant cached

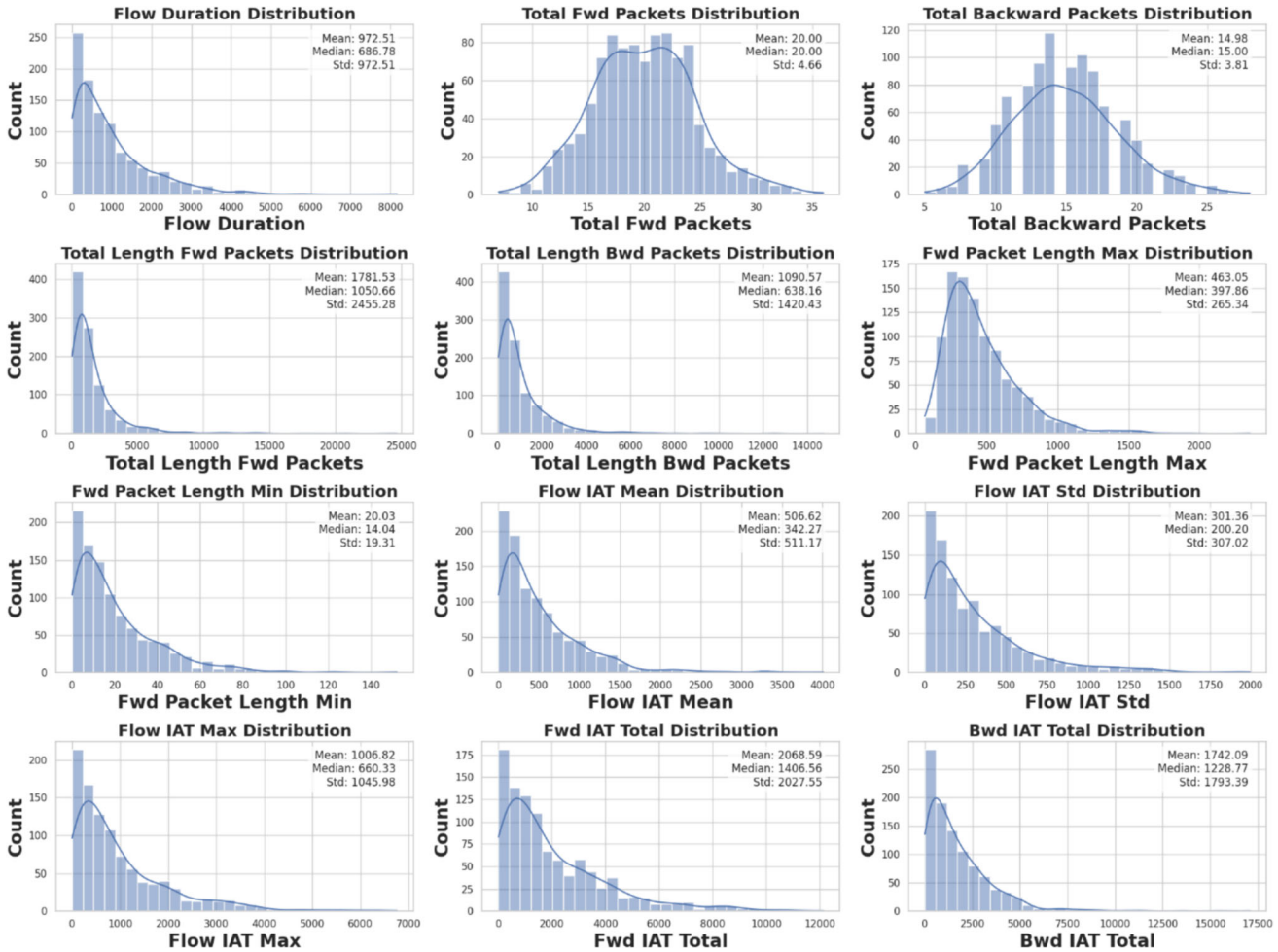

Fig. 9 Visualization analysis of dataset

Table 3 Efficiency Analysis of EDL at different var_{rate} and n_s

var_{rate}	Metrics	EDL	EL	RPCA	QICAN
var_{rate} : 10%	C_{hr}	98.98 ± 0.12	88.37 ± 2.40	91.50 ± 2.34	85.28 ± 2.67
	C_{eff}	97.38 ± 0.22	87.57 ± 2.30	92.45 ± 2.34	87.71 ± 2.73
	E_{fpr}	7.82 ± 0.10	8.35 ± 0.22	8.43 ± 0.03	8.20 ± 0.23
	E_{fnr}	2.34 ± 0.02	10.32 ± 0.12	9.26 ± 2.45	13.70 ± 1.46
	r_d	36.3 ± 0.23	53.89 ± 3.50	47.25 ± 3.23	56 ± 1.34
	err	5.24	8.66	8.24	7.5
var_{rate} : 20%	C_{hr}	99.0 ± 0.12	92.27 ± 2.40	91.25 ± 2.34	86.50 ± 2.67
	C_{eff}	98.65 ± 0.12	91.42 ± 2.40	91.87 ± 2.34	87.30 ± 2.67
	E_{fpr}	8.25 ± 0.10	8.93 ± 0.22	9.02 ± 0.03	8.320 ± 0.23
	E_{fnr}	2.02 ± 0.02	9.68 ± 0.12	8.45 ± 2.45	12.10 ± 1.46
	r_d	34.34 ± 0.23	50.02 ± 3.50	48.2 ± 3.23	50.20 ± 1.34
	err	5.53	8.7	8.64	8.2
var_{rate} : 30%	C_{hr}	99.02 ± 0.12	91.47 ± 2.40	92.93 ± 2.34	87.24 ± 2.67
	C_{eff}	98.76 ± 0.12	91.65 ± 2.40	92.64 ± 2.34	88.43 ± 2.67
	E_{fpr}	8.30 ± 0.10	9.30 ± 0.22	9.15 ± 0.03	8.30 ± 0.23
	E_{fnr}	1.82 ± 0.02	8.25 ± 0.12	6.43 ± 2.45	11.02 ± 1.46
	r_d	31.0 ± 0.23	44.30 ± 3.50	39.3 ± 3.23	48.0 ± 1.34
	err	5.35	9.2	8.23	8.4
var_{rate} : 40%	C_{hr}	98.98 ± 0.23	89.50 ± 0.12	91.20 ± 0.32	85.70 ± 1.46
	C_{eff}	98.34 ± 0.21	90.50 ± 0.12	92.34 ± 0.32	86.30 ± 1.46
	E_{fpr}	8.10 ± 0.02	8.50 ± 0.20	8.90 ± 0.10	8.10 ± 0.10
	E_{fnr}	1.50 ± 0.13	10.20 ± 0.14	8.50 ± 1.45	13.20 ± 2.50
	r_d	30 ± 0.13	43.0 ± 0.23	40 ± 1.20	52.00 ± 1.30
	err	5.16	8.25	8.5	7.38
var_{rate} : 50%	C_{hr}	98 ± 0.24	92.80 ± 0.13	93.10 ± 3.35	88.34 ± 1.50
	C_{eff}	98.34 ± 0.12	91.560 ± 0.13	93.460 ± 3.35	87.84 ± 1.50
	E_{fpr}	7.30 ± 0.01	8.34 ± 0.01	8.53 ± 0.01	8.13 ± 0.23
	E_{fnr}	1.32 ± 0.13	7.24 ± 0.34	7.43 ± 1.40	11.34 ± 3.10
	r_d	30 ± 0.30	42.2 ± 3.00	44.3 ± 0.23	54 ± 0.23
	err	4.5	7.7	7.9	7.1
n_s : 10	C_{hr}	98. ± 0.001	91.80 ± 1.20	94.50 ± 1.40	90.30 ± 1.20
	C_{eff}	98.90 ± 0.30	91.40 ± 2.40	92.90 ± 2.50	88.40 ± 2.40
	E_{fpr}	7.60 ± 0.0002	9.24 ± 0.02	8.30 ± 0.30	8.50 ± 0.30
	E_{fnr}	1.40 ± 0.02	8.60 ± 1.40	7.20 ± 1.40	6.00 ± 2.60
	r_d	31.00 ± 1.20	51.40 ± 2.50	46.00 ± 1.20	38.00 ± 2.10
	err	4.3	9.1	8.91	7.4
n_s : 20	C_{hr}	99.10 ± 0.11	91.70 ± 0.18	94.40 ± 0.25	89.10 ± 0.34
	C_{eff}	98.90 ± 0.12	90.30 ± 0.22	92.70 ± 0.21	88.30 ± 0.33
	E_{fpr}	8.03 ± 0.01	8.56 ± 0.01	8.20 ± 0.01	8.60 ± 0.12
	E_{fnr}	1.54 ± 0.13	8.40 ± 0.22	8.10 ± 0.30	11.60 ± 0.40
	r_d	29.4 ± 0.30	44.00 ± 0.30	33.00 ± 0.23	36.00 ± 0.23
	err	3.54	8.45	7.8	7.3
n_s : 30	C_{hr}	99.20 ± 0.11	91.50 ± 0.18	94.30 ± 0.25	87.30 ± 0.34
	C_{eff}	98.67 ± 0.12	90.10 ± 0.22	93.50 ± 0.21	85.70 ± 0.33
	E_{fpr}	7.20 ± 0.01	8.40 ± 0.01	8.10 ± 0.01	8.30 ± 0.12
	E_{fnr}	1.40 ± 0.13	9.30 ± 0.22	7.30 ± 0.30	11.10 ± 0.40
	r_d	28.00 ± 0.30	39.00 ± 0.30	40.00 ± 0.23	49.00 ± 0.23
	err	3.5	7.8	7.4	8.32

Table 3 (continued)

var_{rate}	Metrics	EDL	EL	RPCA	QICAN
n_s : 50	C_{hr}	99.40 ± 0.10	93.50 ± 0.15	95.210 ± 0.20	88.50 ± 0.30
	C_{eff}	99.10 ± 0.11	91.60 ± 0.20	94.520 ± 0.25	87.30 ± 0.33
	E_{fpr}	7.54 ± 0.01	8.30 ± 0.01	9.31 ± 0.01	8.60 ± 0.12
	E_{fnr}	1.32 ± 0.12	6.46 ± 0.22	7.40 ± 0.30	11.45 ± 0.40
	r_d	27.4 ± 0.30	35.00 ± 0.30	39.00 ± 0.23	52.00 ± 0.23
	err	3.2	6.9	7.2	8.23

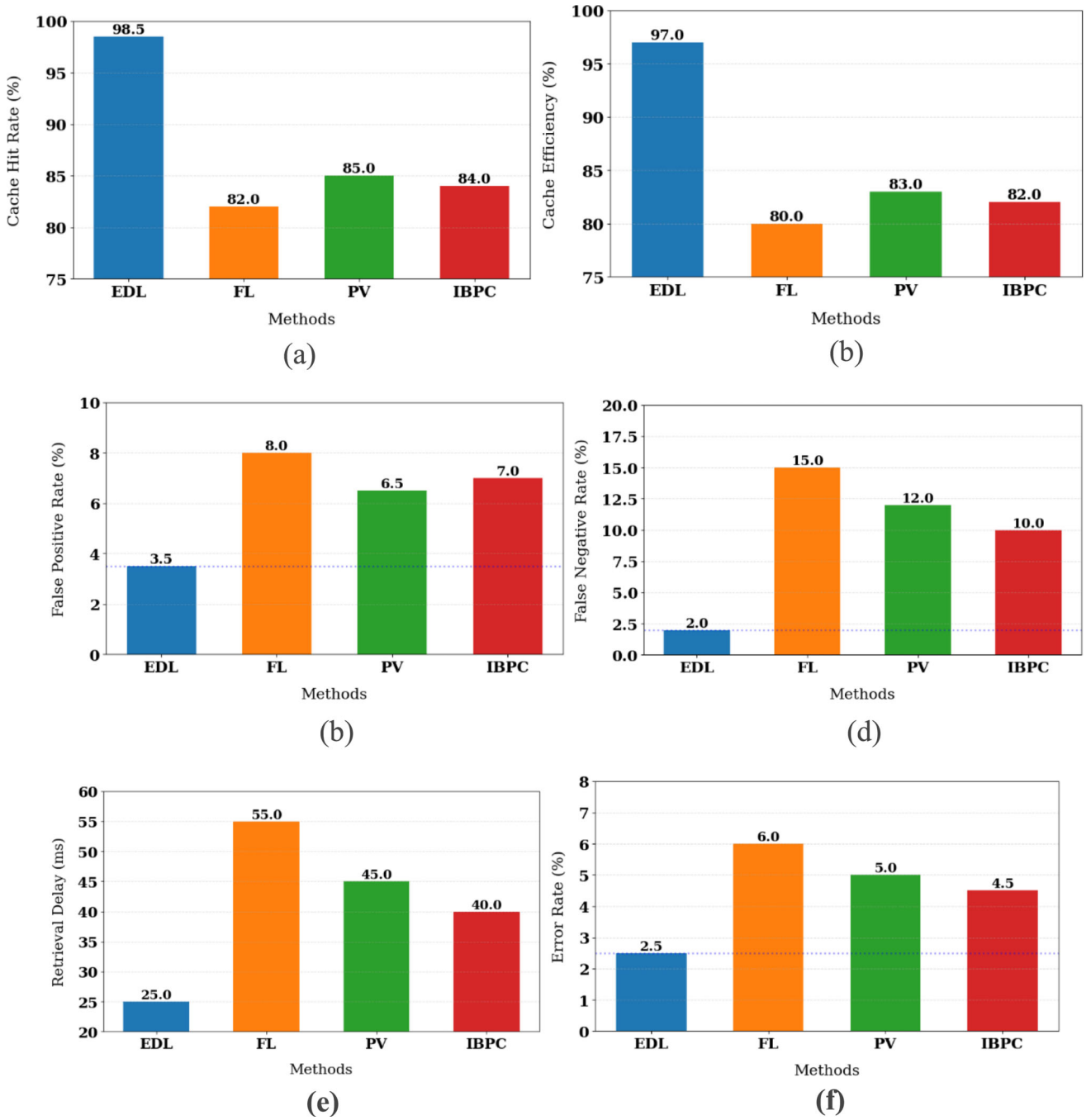

Fig. 10 Efficiency analysis of EDL. **a** Cache hit rate, **b** cache efficiency, **c** false positive rate, **d** false negative rate, **e** retrieval delay, and **f** error rate

content with a 98.5% cache hit rate and 97% cache efficiency, outperforming alternatives (82–85% hit rates). The system's low false positive rate (3.5%) and false negative rate (2%) surpass those of competitors by 2–5 times, demonstrating its accuracy in content identification and retrieval. EDL has the lowest retrieval delay (25 ms), real-time responsiveness, and an ultra-low error rate (2.5%), which is half that of its nearest competitor. The unique deep learning architecture of EDL dynamically adapts to content access patterns while minimizing computing costs to yield these gains. EDL is a reliable option for current caching systems that require accuracy, speed, and efficiency due to its consistent performance gap across all measures, especially error-sensitive measurements. The system's balanced hit optimization (high values) and error minimization (low values) suggest that it resolves the cache responsiveness–reliability trade-off, making it suitable for latency-sensitive applications such as edge computing and content delivery networks. In the present setup, the attack injection rates were evaluated at 10%, 20%, 30%, 40%, and 50% to assess the effectiveness of the proposed deep learning detection system's framework. These discrete levels, while providing a baseline understanding of system performance, fall short of capturing the multifaceted complexities of the NDN world, wherein cache pollution attacks (CPAs) are often executed in erratic, bursty, or sustained low-intensity patterns or through prolonged periods of subdued intensity. This is particularly important, given that system performance does not degrade linearly with the level of attack, as noted in federated learning-based NDN security models [20], the popularity variation mechanism [22], and interface-based popularity control (IBPC) strategies [25]. These works highlight that even slight increases in the rate of attack can disproportionately damage cache optimization, detection accuracy, and processing delays. In support of these conclusions, further extensions of this research should be done using coarser levels of injection (5%, 15%, 25%) alongside temporal burst models that simulate adversarial behavior. In addition, analysis would reveal how detection accuracy and resource efficiency are related to each other, thereby increasing the practical relevance and scope of the proposed framework.

5 Conclusion

Thus, the paper discusses detecting enhanced deep learning (EDL)-based cache pollution attacks in NDN. This study creates effective simulation settings with hardware setup to evaluate the collected CICIDS 2017 dataset information. The traffic preprocessing block is designed with a setup filtering and normalization process that unifies the data, overcoming overfitting issues. Then, the recurrent layer and the memetic optimization technique are applied to predict the malicious and legitimate users with a maximum prediction rate. The network receives input from the preprocessing block that covers every request attribute, such as frequency access request, cache hit rate, temporal pattern, content diversity, and anomaly indicator. These features help to identify how effectively the system recognizes the irrelevant access and provides an effective cache management process. During the classification process, memetic operators such as selection, crossover, and mutation identify the best solution by balancing the exploitation and exploration. The selected parameter balances the system's robustness and flexibility while sending requests in the NDN environment. The EDL approach attains a 3.2 error rate and 98–99% cache hit rate from the developed system compared to other methods. The variability rates of various attacks and the network size determine the system's effectiveness. Although the system effectively processes the different features when handling high-dimensional data, it incurs a high computation overhead. Therefore, the system requires optimization techniques in the future to reduce the feature subset and manage computation overheads by up to 10%.

Data availability None

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Anjum A, Agbaje P, Mitra A, Oseghale E, Nwafor E, Olufowobi H (2024) Towards named data networking technology: emerging applications, use cases, and challenges for secure data communication. Future Gener Comput Syst 151:12–31
2. Khalid A, Rehman RA, Kim BS (2024) Caching strategies in NDN based wireless ad hoc network: a survey. Comput Mater Contin. https://doi.org/10.32604/cmc.2024.049981
3. Alshorman M, Saif Taleb (2025) Cluster-based traffic management for optimizing urban congestion using unsupervised learning on real-time data streams. PatternIQ Mining 2(1):24. https://doi.org/10.70023/sahd/250206
4. Shah MSM, Leau YB, Anbar M, Bin-Salem AA (2023) Security and integrity attacks in named data networking: a survey. IEEE Access 11:7984–8004
5. Wang J, Wei X, Fan J, Duan Q, Liu J, Wang Y (2023) Request pattern change-based cache pollution attack detection and defense in edge computing. Digit Commun Networks 9(5):1212–1220
6. Zhang Z, Lung CH, Wei X, Chen M, Chatterjee S, Zhang Z (2023) In-network caching for ICN-based IoT (ICN-IoT): a comprehensive survey. IEEE Internet Things J 10(16):14595–14620
7. Babu VJ (2024) Malicious source detection and threats mitigation in named data networking using deep learning. Int J Intell Eng Syst 17(5):440
8. Magsi AH, Yovita LV, Ghulam A, Muhammad G, Ali Z (2023) A content poisoning attack detection and prevention system in vehicular named data networking. Sustainability 15(14):10931
9. Dong J, Wang K, Quan W, Yin H (2020) Interestfence: simple but efficient way to counter interest flooding attack. Comput Secur 88:101628
10. Yao L, Zeng Y, Wang X, Chen A, Wu G (2020) Detection and defense of cache pollution based on popularity prediction in named data networking. IEEE Trans Depend Secure Comput 18(6):2848–2860
11. Yao L, Chen Z, Dai H, Wu G (2021) Exploiting non-cooperative game against cache pollution attack in vehicular content centric network. IEEE Trans Depend Secure Comput 19(6):3873–3886
12. Vasudevan VA, Tselios C, Politis I (2020) On security against pollution attacks in network coding enabled 5G networks. IEEE Access 8:38416–38437
13. Reali G, Femminella M (2021) Two-layer network caching for different service requirements. Future Internet 13(4):85
14. Dong C, Wang H, Ni D, Liu Y, Chen Q (2020) Impact evaluation of cyber-attacks on traffic flow of connected and automated vehicles. IEEE Access 8:86824–86835
15. Ali TE, Chong YW, Manickam S (2023) Machine learning techniques to detect a DDoS attack in SDN: a systematic review. Appl Sci 13(5):3183
16. Mittal M, Kumar K, Behal S (2023) Deep learning approaches for detecting DDoS attacks: a systematic review. Soft Comput 27(18):13039–13075
17. Yao L, Zheng Z, Wang X, Zeng Y and Wu G (2023) Detection of cache pollution attack based on ensemble learning in ICN-based VANET. In: IEEE transactions on dependable and secure computing, 20(4):3287–3298. https://doi.org/10.1109/TDSC.2022.3196109.
18. Hidouri A, Touati H, Hadded M, Hajlaoui N, Muhlethaler P, Bouzefrane S (2023) Q-ICAN: a Q-learning based cache pollution attack mitigation approach for named data networking. Comput Netw 235:109998
19. Kim H, Hahn C, Kim HJ, Shin Y, Hur J (2023) Deep learning based detection for multiple cache side-channel attacks. IEEE Trans Inf Forensic Secur 19:1672–1686
20. Yao L, Li J, Deng J, Wu G (2023) Detection of cache pollution attack based on federated learning in ultra-dense network. Comput Secur 124:102965
21. Kar P, Chen L, Sheng W, Kwong CF, Chieng D (2024) Advancing NDN security: efficient identification of cache pollution attacks through rank comparison. Internet of Things 26:101142
22. Sameer MK, Salman MI (2023) Detection and mitigation of cache pollution attack using popularity variation in information centric networking based on SDN. Iraqi J Sci 64(3):1442–1462
23. Yao L, Zheng Z, Wang X, Zeng Y, Wu G (2022) Detection of cache pollution attack based on ensemble learning in ICN-based VANET. IEEE Trans Depend Secure Comput 20(4):3287–3298
24. Zou Q, Zhang L, Singhal A, Sun X, Liu P (2024) Analysis of neural network detectors for network attacks. J Comput Secur 32(3):193–220

25. Kumar N, Srivastava S (2024) IBPC: an approach for mitigation of cache pollution attack in NDN using interface-based popularity. Arab J Sci Eng 49(3):3241–3251
26. https://www.kaggle.com/datasets/kk0105/cicids2017

Authors and Affiliations

S. Sethu[1] · A. Manikandan[1]

✉ S. Sethu
sethu.se@vistas.ac.in

[1] Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Tamilnadu, India