

Next-generation IIoT security: Comprehensive comparative analysis of CNN-based approaches

Huiyao Dong^a, Igor Kotenko^b, Dmitry Levshun^{b,*}

^a ITMO University, Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia

^b St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS), 39, 14th Liniya, St. Petersburg, 199178, Russia

ARTICLE INFO

Keywords:

Information security
Artificial intelligence
Deep learning
State assessment
Attack detection
Network traffic
Industrial IoT

ABSTRACT

Industrial Internet of Things (IIoT) presents a range of benefits but also introduces security vulnerabilities. This paper systematically compares different deep learning model structures for IIoT intrusion detection. Four Convolutional Neural Network (CNN) classifiers are implemented, including hybrid CNN-GRU, 1D Xception, and 1D Resnet models. The evaluation focuses on robustness and generalizability across two datasets to assess overall performance and detection rates. To handle imbalanced data, preprocessing involves PCA feature selection and hybrid resampling techniques. Furthermore, we design a distributed training process for massive datasets and continuous learning, enabling efficient large-scale processing. For the experimental evaluation, we make use of two datasets with multi-label classification tasks and analyze performance metrics of different models in terms of their detection abilities and efficiency. The proposed methods demonstrate strong performance, successfully identifying even rare attacks. In addition, we conduct a performance comparison with existing publications that utilize the same datasets, confirming that our models are on par with state-of-the-art IIoT models.

1. Introduction

Industrial Internet of Things (IIoT) leverages modern information, communication, and management technologies to create smart industrial environments [1]. By enabling autonomous data analytics, equipment failure detection, operational optimization, and seamless communication, IIoT has become essential for businesses, driving smart manufacturing, enhanced decision-making, and data analysis. It also plays a critical role in monitoring and managing vulnerable infrastructures like smart cities, healthcare, intelligent transportation systems, and device-to-device communication [2]. The rapid growth of IIoT is evident in its industrial applications, where it excels in reliable and efficient operations. However, while IIoT automates processes, it introduces privacy and security challenges for current and future systems [3]. Alongside stringent deadlines, security breaches can result in significant safety and economic consequences. Furthermore, although IIoT devices generate vast amounts of data, resource constraints and the heterogeneity of smart devices make them vulnerable to threats [4].

Given these risks, implementing effective security measures is critical. Over the past decade, intelligent techniques, particularly machine learning (ML), have been widely used. Traditional ML and statistical

models remain effective, especially in supervised learning tasks, including AdaBoost, K-nearest Neighbors (KNN), Random Forest (RF), Support Vector Machines (SVM), Association Rules, and Decision Trees (DT) [5]. Recently, there has been a shift toward deep learning (DL) models in IIoT security. Research focuses include leveraging deep feed-forward neural networks (DNNs) and Convolutional Neural Networks (CNNs) for enhanced performance [6], exploring hybrid models like CNN + Long Short-Term Memory (LSTM) [7], and employing transfer learning and pretraining strategies to improve intrusion detection with limited labeled data.

While many hybrid models using CNN have shown promising results, open questions remain about model architecture choices: Does model depth positively correlate with performance? Can dynamic blocks like Xception and Residual outperform regular convolutional blocks? Prior research has investigated the integration of CNN architectures with multitask learning (MTL), showcasing the efficacy of CNN approaches [8]; however, this study primarily delved into the exploration of the MTL framework, with limited discussion on CNN architectures per se and hybrid models incorporating CNN. To investigate these questions associated with CNN usage, we systematically compare various CNN structures on IIoT intrusion detection tasks.

* Corresponding author.

E-mail addresses: hydong@itmo.ru (H. Dong), ivkote@comsec.spb.ru (I. Kotenko), levshun@comsec.spb.ru (D. Levshun).

The primary novelty of this study lies in the development, optimization, and extensive evaluation of four distinct CNN-based architectures specifically designed to address the challenges of sequential network traffic data analysis and imbalance in IIoT datasets. These contributions aim to enhance detection capabilities for rare and evolving network threats, which is critical in IIoT systems and can be summarized as follows:

1. Novel CNN-based classifiers tailored for IIoT: We designed and implemented four innovative CNN-based architectures that cater to the unique demands of sequential network traffic analysis. These include (1) a deep CNN model; (2) hybrid CNN-GRU architectures introducing temporal analysis for sequential patterns; and (3) adaptations of advanced one-dimensional Xception and ResNet architectures refined for IIoT contexts. This diversity allows us to address varying types of network data while enhancing detection efficiency and scalability.
2. Comparative architectural analysis: A comprehensive structural comparison is conducted for our models, investigating key design elements such as network depth, convolutional block configurations (Convolutional, Xception, and Residual), and model performance under different CNN configurations. This unique analysis offers insights into how specific architectural elements influence detection capabilities and efficiency for sequential traffic data.
3. Hybrid resampling technique tailored for IIoT applications: in response to the inherent data imbalance, we introduce an innovative approach combining an enhanced random undersampling method with a Generative Adversarial Network (GAN)-based oversampling technique to effectively mitigate the skewed distribution of network traffic data. Both methods are designed to be adaptive and robust for handling diverse and evolving network scenarios.
4. Robustness and generalizability evaluation: the effectiveness of these models is rigorously tested across two different datasets to assess their robustness and generalizability. Performance metrics focus not only on overall efficacy, but also on the detection rate of rare attacks, demonstrating that our CNN-based approaches are not only competitive but also excel in identifying elusive security threats.

The rest of this paper is organized as follows. Section 2 provides a comprehensive review of related work in CNN-based network architectures and methods used to address imbalanced datasets. Section 3 introduces the proposed approach, which focuses on utilization of different CNN-based structures. Section 4 presents the experimental results, while also comparing the suggested approach with other state-of-the-art methods. The discussion section delves into the main findings and identifies potential areas for future research. Finally, we conclude the paper.

2. Related work

Deep learning (DL) has gained considerable attention and been extensively applied in the IIoT security domain. One prominent research direction involves integrating computer vision (CV)-inspired models. For example, Wen et al. [9] employed LeNet-5 for fault diagnosis by transforming IIoT data into 2-D matrices, aligning with an image classification approach. Another study used a multi-CNN fusion algorithm for intrusion detection, enhancing security mechanisms through ensemble CNN models that significantly outperformed traditional methods [10]. Beyond CNNs, other DL models have also been explored for IIoT security. Dehlaghi-Ghadim et al. [11] proposed an IDS based on a feed-forward artificial neural network (ANN), demonstrating the impact of optimization algorithms, with experiments achieving an accuracy of 0.995. Similarly, a DNN-based IDS framework was introduced in [12]

for the CICIOT2023 dataset, where different layers and units were analyzed. Another study [13] compared a basic neural network with ensemble learning methods, utilizing a genetic algorithm for optimization. While accuracy exceeded 0.99, the detection rate was only around 0.80. Additionally, hybrid models combining different types of layers have been investigated. For instance, a comparative analysis [14] implemented an IDS model with only CNN layers and a hybrid model combining CNN and LSTM layers, achieving an overall accuracy of 0.95.

Another research trend addresses privacy concerns. Federated Learning (FL), employing both synchronous and asynchronous methodologies, offers flexibility to tackle challenges posed by the large volume and diversity of devices in IoT environments [15]. Yazdinejad et al. [16] proposed a blockchain-based IIoT security solution, emphasizing the use of FL in the Internet of Vehicles to develop a global machine learning model for anomaly detection. Another notable area focuses on security solutions operating within blockchain frameworks. For instance, a recent IoT security framework [17] introduced enhanced fuzzy DL models for threat detection, along with an intelligent fuzzy layer for real-time processing and privacy-preserving security management. The framework also included techniques to handle low-quality data, improving model accuracy.

Efforts to improve classifier performance have led researchers to explore data processing techniques for enhancing the quality of IIoT data. Gaber et al. [18] evaluated various feature selection algorithms, including particle swarm optimization (PSO) and simulated annealing (BAT), using three well-known classifiers—RF, KNN, and Multi-layer Perceptron (MLP)—for performance assessment. Similarly, Zolanvari et al. [19] tackled challenges in IIoT security analysis, particularly imbalanced data, by adjusting the proportion of malicious behaviors in network traffic data (ranging from 0.1% to 1%) and evaluating artificial neural networks using multiple metrics. Another focus is undersampling techniques, designed to reduce the number of benign samples in network traffic, thereby mitigating data imbalance and improving training efficiency. Advanced techniques often employ the KNN algorithm, such as Edited Nearest Neighbors (ENN) [20], which removes samples near the decision boundary, and Tomek Links [21], which eliminates pairs of samples to better define class boundaries. However, KNN-based methods require considerable processing time, especially with larger datasets, higher values of K , or numerous classes, making them impractical for resource-constrained devices. Alternatives include hybrid models to enhance feature learning [22] or the integration of Edited Nearest Neighbor undersampling algorithms with multitask learning frameworks using optimized task-specific loss weights [23].

The selected research focuses on deep learning, anomaly detection, intrusion detection, and privacy-preserving security solutions in IIoT systems, with trends of transitioning to deep learning and addressing imbalanced classification and efficient network data processing overflow. Table 1 presents a summary of the cited work with a focus in DL approaches, focusing on their main model, feature selection and oversampling techniques used, and whether there is architecture optimization discussion and comparative analysis.

Previous research has explored various DL architectures for enhancing IIoT security. However, gaps remain regarding the development of a simple yet efficient undersampling method to address class imbalance and an advanced oversampling technique capable of efficiently generating realistic data. Additionally, questions about optimal model depth, convolutional block selection, and layer design are yet to be fully addressed. A comprehensive comparative analysis of popular CNN models with different architectures, as well as hybrid approaches incorporating GRU, is necessary to provide insights into the strengths and limitations of CNN-based methods, guiding future research toward optimized architectures for IIoT security. While existing research has utilized SMOTE and other KNN-based adaptive oversampling methods, we propose a GAN-based adaptive oversampling technique to address data imbalance. Our GAN framework incorporates DNN layers with a shallow architecture to ensure practicality in IIoT environments.

Table 1
Comparative analysis of DL-based intrusion detection approaches.

Paper	Models used	Dimension reduction	Undersampling	Oversampling	Architecture optimization	Comparative study	Robustness evaluation
[10]	Clustering+Multi-CNN Fusion	No	No	No	No	Yes	No
[11]	ANN	MRMR	No	No	No	No	No
[12]	DNN, CNN1/2/3D, CNN+LSTM	No	No	No	No	Yes	Yes
[14]	DNN	CFS, MRMR, RFE	No	No	Yes	No	No
[22]	CNN	Kernel PCA	No	No	No	No	Yes
[23]	CNN AutoEncoder	ENR	ENR	SMOTE	No	No	No
[24]	CNN	GWO	No	No	No	No	Yes
[25]	DNN, LSTM	RF FI	Random	SMOTE	No	No	No
Ours	4 CNN-based variations	PCA	Adaptive random	Adaptive GAN	Yes	Yes	Yes

MRMR – Maximum Relevance and Minimum Redundancy; CFS – Correlation-based Feature Selection; RFE – Recursive Feature Elimination; ENR – Elastic Net Regularization; GWO – Grey Wolf Optimization; RF FI – Random Forest-based Feature Importance.

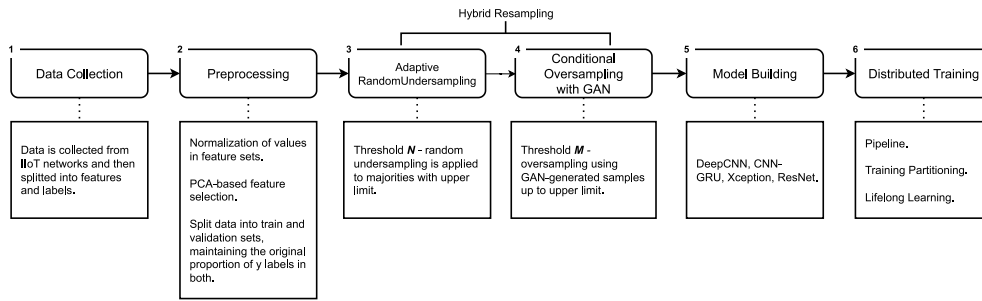


Fig. 1. Overall workflow.

3. Methodology

Fig. 1 illustrates the workflow of our methodology, which employs a custom random undersampling, an adaptive GAN-based oversampling and Convolutional Neural Network (CNN)-based multi-label classifiers.

First, the feature dataset X is standardized to ensure uniformity by removing the mean and scaling the data to have unit variance. The dataset is subsequently divided into training and validation sets to facilitate effective training and evaluation of the models.

To address class imbalances, we firstly apply a custom random undersampling on the training set. In this step, each majority class is undersampled to a predefined number N (e.g., 50 000 samples), while minority classes with initial sample sizes below N remain untouched to retain their original distribution. Following undersampling, we employ GAN-based oversampling techniques to generate synthetic samples for minority classes. This step increases the sample volume of each class to a threshold M (e.g., 10 000 samples).

After preprocessing, the standardized data is used to train the CNN models developed for this study. These models are specifically designed to handle the structure and complexity of multi-label classification tasks. Finally, the performance of the classification models is assessed using various metrics, with a particular focus on the detection ability for minority classes. This comprehensive evaluation ensures that the effectiveness of the model is reliably measured, especially in identifying underrepresented classes.

3.1. Data collection

IIoT networks generate heterogeneous data streams from distributed sensors, edge devices, and network traffic. The primary data streams are the network traffic data, namely the raw packets captured using network taps. The included attributes, such as flow duration, protocol

types, packet rates, and flag counts, can form the basis for attack detection. To create a structured and labeled dataset, the raw data require multiple preprocessing steps:

1. **Normalization:** sensor and network metrics should be scaled (min-max or z-score) for compatibility.
2. **Label Assignment:** binary/multi-class labels are mapped to network traffic based on ground-truth.
3. **Data Fusion:** network and device data are merged into a unified feature matrix using timestamps or session identifiers.

3.2. Preprocessing

IIoT data often contains redundancies and noise, which can hinder real-time analysis. To address this, our preprocessing pipeline consists of three key steps: data standardization, PCA-based dimensionality reduction, and data splitting. First, raw data are standardized to ensure consistent scales across features, a critical step for effective PCA execution. Next, PCA is applied to reduce the dimensionality of the dataset by identifying components that maximize variance while minimizing information loss. Finally, the dataset is split into training and testing sets to ensure unbiased evaluation of the proposed models. The rest of this section details the PCA-based dimensionality reduction process, which is central to our preprocessing workflow.

PCA is a variance-maximizing orthogonal transformation and can be applied in the dimension reduction task. Let the raw dataset be represented as $X \in \mathbb{R}^{n \times p}$, where n = samples and p = features. PCA identifies a projection matrix $W \in \mathbb{R}^{p \times k}$ ($k < p$) that meets the following [26]:

$$W^* = \arg \max_W \text{Var}(XW) \quad \text{subject to} \quad W^T W = I \quad (1)$$

In which w refers to each column of the projection matrix W , namely, one principal component. Here, W 's columns (principal components) maximize retained variance (signal preservation) while enforcing $W^T W = I$. In the implementation, we configure PCA with a 95% explained variance threshold, empirically balancing dimensionality reduction and information loss.

We adopt PCA in the proposed workflow, as its ability to reduce dimensionality while retaining maximum data variance is critical for IIoT data. By prioritizing a comprehensive variance explanation, PCA ensures that classification models receive the most relevant information, enhancing classification accuracy. Additionally, PCA improves interpretability by providing the accumulated variance explanation ratio, which quantifies the information retained by each principal component and highlights the contribution of each dimension to the data structure. Using a 0.95 explained variance threshold, we determine the optimal number of components, ensuring dimension reduction does not compromise data integrity. In summary, PCA enhances model performance while contributing to the transparency and adjustability of the feature selection process, supporting robust and explainable analysis in IIoT environments.

3.3. Adaptive random undersampling

Imbalanced classification presents a significant challenge in IIoT-focused network security analysis, as benign behaviors and common attacks (e.g., Denial of Service, DoS) dominate network traffic, while rare but critical malicious attacks represent only a small fraction of the data. To address this imbalance and enhance training efficiency in multi-label classification tasks, we propose an adaptive undersampling module, outlined in Algorithm 1.

This module operates by managing the data distribution for each label through a flexible target count parameter, which defines the maximum number of samples allowed per class. The key steps include:

1. *Class Distribution Evaluation*: a dictionary, *class counts*, is initialized to track the frequency of each class. The algorithm iterates through the dataset per class.
2. *Randomized Selection Based on Distribution*: for any class, where *class counts* exceeds the predefined *target count*, the algorithm performs random undersampling to reduce samples until the limit is satisfied.
3. *Parameter Flexibility*: the *target count* parameter can be adjusted based on specific characteristics of the dataset or system constraints, ensuring scalability to its various sizes and application scenarios.

This modular design ensures that the undersampling process is adaptable to diverse IIoT use cases, enabling efficient handling of imbalanced data. The module is particularly suited for environments requiring rapid, real-time processing, such as security monitoring systems where identifying rare yet dangerous attacks is critical. Furthermore, its lightweight implementation supports the deployment on resource-constrained IIoT devices.

3.4. Conditional oversampling with GAN

Imbalanced datasets in IIoT-focused security systems often suffer from a scarcity of rare attack samples, which limits classifier effectiveness. To mitigate this challenge, we propose the ConditionalGANOverSampler, a GAN-based technique for generating synthetic samples to balance class distributions. The method combines a conditional autoencoder (AE) with adversarial training to produce realistic data for underrepresented classes. Fig. 2 illustrates the GAN framework and data flow, highlighting its two main components: the generator and the discriminator.

The generator is designed to create synthetic samples representative of the rare class distributions. For this, we employ a DNN-based

Algorithm 1 RandomUndersampling

```

X — Feature dataset
y — Class labels
class_counts ← Dictionary to store the count of each class
undersampled_X = [] ← Undersampled feature dataset
undersampled_y = [] ← Undersampled class labels
target_count = 10000 ← The desired up-limit for each class
after undersampling

Check each class's portion count and apply random sampling
if count > target_count

  for label in unique(y) do
    if class_counts[label] > target_count then
      indices ← Indices of samples with the current label in
      y
      undersampled_indices ← Randomly select target_count
      indices
      undersampled_X.append(X[undersampled_indices])
      undersampled_y.append(y[undersampled_indices])
    else
      undersampled_X.append(X[y == label])
      undersampled_y.append(y[y == label])
    end if
  end for

Return undersampled_X, undersampled_y

```

conditional AE due to its lightweight architecture, which is suitable for resource-constrained IIoT devices. Input data is passed through fully connected layers with ReLU activation and a dropout layer (0.1 dropout rate). A latent layer condenses the input into a bottleneck representation to capture key features. The encoded representation is concatenated with class labels to guide synthetic data generation. The decoder reconstructs data by feeding concatenated inputs into DNN layers (final layer uses a sigmoid activation to scale output). The generator is pretrained on real data to initialize weights, stabilizing subsequent adversarial training. This setup allows the conditional setting to enhance representation of rare classes while maintaining computational efficiency.

The discriminator evaluates the realism of the samples generated by the generator; built with multiple DNN layers, it processes both real and generated data batches. The output is a binary classification, predicting whether a given input is real or synthetic. A binary cross-entropy loss function ensures the discriminator becomes adept at distinguishing generated samples from real ones. The implementation logic of ConditionalGANOverSampler follows Algorithm 2. It operates in three key phases:

1. *Pretraining the Autoencoder*: during initialization, the generator is pretrained on the real dataset. This step stabilizes GAN training by enabling the generator to start from well-structured data representations rather than random weights.
2. *Adversarial Training*: synthetic samples are generated by conditioning the decoder on predefined class labels. Alternating training occurs, where the discriminator learns to classify data as real or generated, and the generator improves to fool the discriminator. The discriminator's weights are frozen during synthetic data generation to prioritize generator optimization.
3. *Adaptive Oversampling*: the *oversample* method adaptively generates synthetic samples for underrepresented classes until each meets a predefined threshold. This ensures class distribution is balanced while minimizing unnecessary data generation for dominant classes.

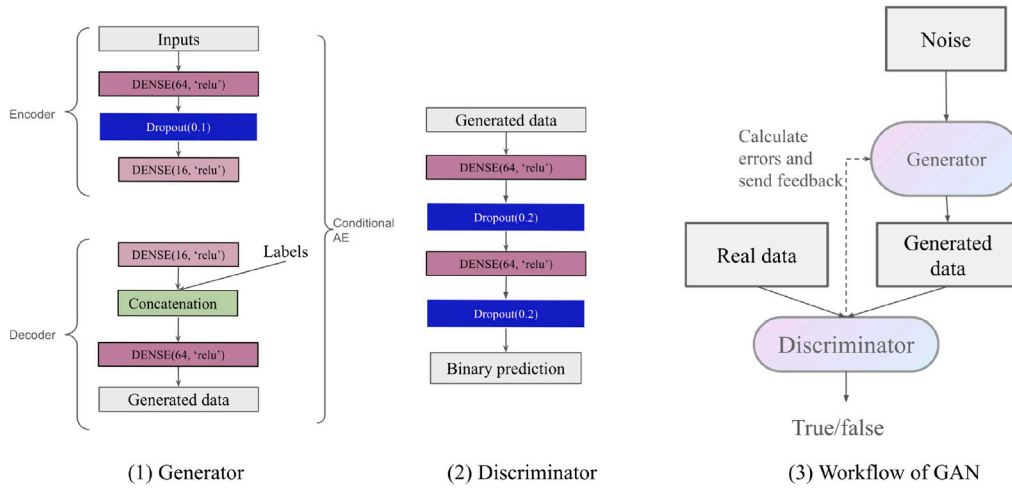


Fig. 2. GAN framework: Model architecture of Generator, Discriminator, and Data Flow within the framework.

Algorithm 2 ConditionalGANOversampler

```

Initialize models: decoder, autoencoder, discriminator
Initialize counting threshold of each class  $threshold = 10000$ 
Initialize pretraining loop  $pretrain = 100$ 
Compile autoencoder and train on  $(x_{train}, y_{train})$  for  $pretrain$  epochs
Method: train_gan
for  $epoch$  in range( $epochs$ ) do
    Generate random noise samples
    Use decoder to generate fake samples (gen_data) conditioned on  $y_{train}$ 
    Create labels for real (valid) and fake (fake) samples
    Train with a batch of real samples and update generator's weights
    Train gan with noise samples and  $y_{train}$  to fool discriminator
    if  $epoch \% 10 == 0$  then
        Print discriminator and generator losses for monitoring purpose
    end if
end for
Method: oversample
Identify unique classes and their counts in  $y_{train}$ 
Initialize empty lists:  $x_{new}, y_{new}$ 
for each class and its count do
    if count < threshold then
        Print number of samples needed for this class
        Generate random noise samples
        Create labels for current class
        Use decoder to generate synthetic data conditioned on noise and class labels
        Append generated data to  $x_{new}$  and labels to  $y_{new}$ 
    end if
end for
Concatenate new data with original training data to get balanced dataset  $(x_{train}, y_{train})$ 
Return oversampled  $x_{train}$  and  $y_{train}$ 

```

By combining conditional settings with adversarial training, ConditionalGANOversampler generates realistic, high-quality samples that effectively address the class imbalance. Its lightweight architecture and ability to utilize pretrained weights make it highly suitable for real-time IIoT systems, particularly in resource-constrained environments. The module enhances the detection of rare attacks and contributes to more robust security solutions.

3.5. Model building

To address the challenge of analyzing one-dimensional sequential data in IIoT applications, we propose and utilize diverse CNN-based modules tailored for hierarchical feature extraction and effective sequence modeling. Fig. 3 presents the general architecture of the proposed model alongside its core CNN-based building blocks (Conv, Xception, Residual).

Within the unified framework, these blocks are sequentially integrated following the input processing layer to enable hierarchical feature extraction. To systematically highlight the relationship between individual components and their implementation within hybrid architectures, this figure only presents the CNN blocks to be adopted in the framework presented in (a). The Conv-GRU hybrid model—which employs these blocks in an integrated, non-modular configuration—is detailed in Fig. 4.

After flattening the feature extraction output from specific convolutional blocks, a dropout layer with a 20% rate is applied to reduce overfitting by randomly deactivating a portion of neurons. Following this, 3 fully connected feed-forward layers comprising 128 units each are utilized for further feature learning. Lastly, an output layer is employed to generate the final predictions.

The final layer yields class predictions based on the corresponding activation function: for binary classification, the activation will be a Sigmoid function; while for multi-label classification – Softmax function. The choice of loss function for output layers of all models depends on the number of classes: categorical cross-entropy is used for multiple classes; and binary cross-entropy is used for binary classification tasks.

The following subsections detail the specific CNN blocks designed for this framework.

3.5.1. Convolutional block

Increasing the depth of CNN models is widely regarded as an effective way to enhance classifier performance. The progression from LeNet5 to AlexNet and VGG demonstrates a trend toward deeper architectures. The primary advantage is the ability to learn more complex features from input data [27]. However, simply increasing depth has its limitations. Deeper networks are more prone to overfitting, leading to high variance and reduced generalization. Additionally, deeper models require more training time and computational resources, making them less practical for classifiers in IIoT devices. Designed for flexibility, the convolutional block consists of:

1. Two convolutional layers, each with identical units, followed by ReLU activation for non-linear transformations.

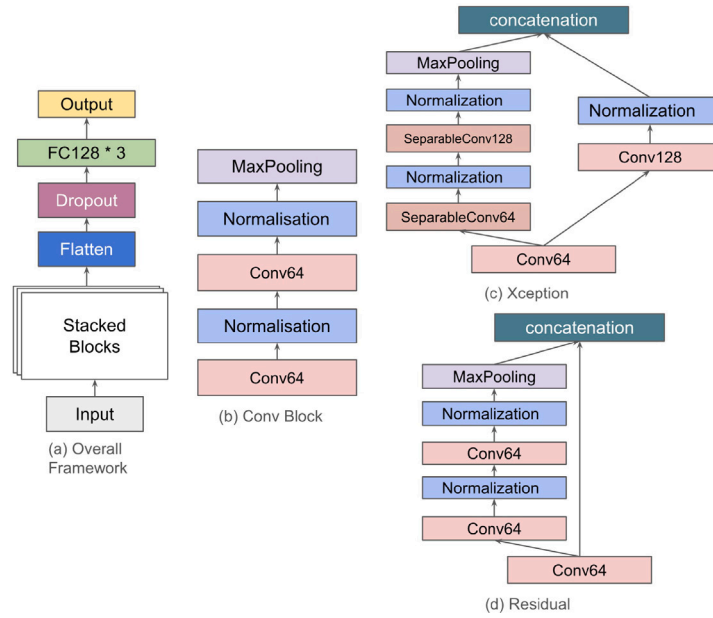


Fig. 3. Core building blocks of the hybrid model. Subfigures (b-d) represent standardized CNN-based modules used in the general architecture (a).

2. *Batch normalization* after each layer to stabilize training and accelerate convergence.
3. *Max-pooling* with a size of 2 to reduce the spatial dimensions of feature maps and retain relevant patterns.

This module enables the extraction of complex patterns while optimizing computational resources, making it suitable for resource-constrained devices.

3.5.2. One dimensional Xception

The Xception Block presented Fig. 3(c) extends the Conv Block by employing separable convolutional operations for efficient feature extraction. Inspired by the Inception [28] architecture, it involves:

1. *Branching Structure*: one branch employs separable convolutions (SeparableConv1D) with a kernel size of 3 and batch normalization applied after each layer. Another branch includes a single convolutional layer followed by normalization.
2. *Parameter Scaling*: units in stacked Xception blocks follow this rule: if $\text{units} \times 2 > 256$, then $\text{units} = 256$; otherwise, $\text{units} = \text{units} \times 2$.
3. *Output Combination*: outputs from both branches are merged using element-wise addition to enhance feature representation.

This structure minimizes parameter overhead while preserving modeling capacity, making it computationally lightweight for IIoT applications.

3.5.3. One dimensional resnet

The residual learning firstly proposed for solving the degradation issue and enhancing the network's learning capability [29]. The Residual Block addresses the limitations of deep networks, such as vanishing gradients, by introducing skip connections. Its structure, see Fig. 3(d), includes:

1. *Main Branch*: two convolutional layers (64 filters, kernel size = 3) followed by batch normalization.
2. *Skip Connection*: the input tensor is directly added to the output of the convolutional sequence using element-wise addition to form residual mappings.

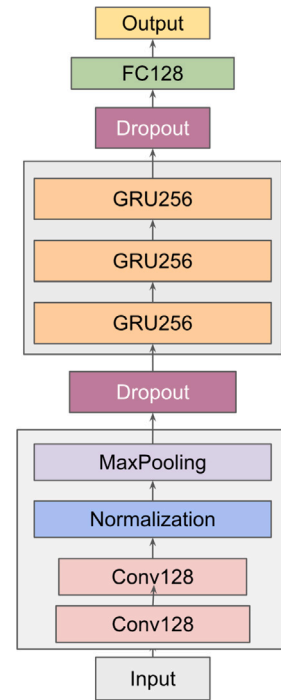


Fig. 4. Core building blocks of the hybrid model. Subfigures (b-d) represent standardized CNN-based modules used in the general architecture (a).

The soft convolutional operation in the block merges the representation learned by main branches with the original block input and allows the network to learn residual functions, making it easier to optimize and reducing the risk of a gradient as the network depth increases.

3.5.4. CNN-GRU hybrid model

The CNN-GRU hybrid model integrates CNN-based feature extraction and GRU-based temporal modeling to capture both spatial and sequential dependencies in one-dimensional data. Fig. 4 illustrates the architecture, which includes:

Table 2
Edge IIoT dataset.

Attack type	Count	Share, %	Attack type	Count	Share, %
Normal	1 615 643	72.830	Uploading	37 634	1.696
DDoS_UDP	121 567	5.480	Backdoor	24 862	1.121
DDoS_ICMP	116 436	5.249	Port_Scanning	22 564	1.017
SQL_injection	51 203	2.308	XSS	15 915	0.717
Password	50 153	2.261	Ransomware	10 925	0.492
Vulnerability_scanner	50 110	2.259	Fingerprinting	1001	0.045
DDoS_TCP	50 062	2.257	MITM	400	0.018
DDoS_HTTP	49 911	2.250			

1. **CNN Block:** two convolutional layers (128 filters) with normalization and max-pooling layers for hierarchical feature extraction. Dropout (20% rate) for regularization.
2. **GRU Layers:** three GRU layers, each with 256 units, are employed to capture temporal relationships in the data, mitigating the vanishing gradient problem inherent to RNNs.

By combining the strengths of both CNNs and GRUs, this hybrid model provides a robust framework for sequential data analysis, particularly in IIoT scenarios where efficiency and accuracy are critical.

3.6. Distributed training strategy

The increasing availability of large datasets requires efficient and scalable methods for model training. Distributed and continuous training is a crucial approach to tackle the challenge posed by large data sets and the necessity for ongoing model refinement with new data. To address IIoT devices' computational constraints while handling large-scale, evolving security datasets, we propose a structured training framework comprising the following core modules. The implementation utilizes TensorFlow [30] (tf) for tasks such as dataset management, batch processing, data conversion, partitioning, and continuous learning.

- **Modular Data Pipeline:**
 - Data Distribution Strategy: Horizontal partitioning of input streams into batches.
 - Framework-Specific Optimization: native TensorFlow Dataset conversion (via `tf.data.Dataset`) for hardware-aware prefetching and parallelization.
- **Adaptive Training Partitioning:** dynamic splits using `tf.distribute.MirroredStrategy` to preserve temporal coherence in continuous IIoT data flows.
- **Lifelong Learning Adaptation:** incremental model refinement via momentum-based retraining (learning rate $\eta_t = \eta_0/\sqrt{t}$) with rolling data windows.

4. Experimental evaluation

4.1. Datasets description

Edge-IIoT [31] is a realistic cybersecurity dataset suitable for ML-IDS. The dataset includes 61 correlated features collected through 7 layers, such as Cloud Computing, IoT and IIoT Perception Layer. It identifies 14 attacks related to connectivity protocols, while 72.83% of samples are normal traffic behaviors. Table 2 outlines the share of different types of traffic.

CICIoT2023 [32] is another novel IIoT attack dataset, involving 44 features and 33 attacks in an IIoT topology of 105 devices. These attacks are classified into seven categories, including DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai. As presented in Table 3, there exists extremely minor classes with this share of lower than 0.1%.

4.2. Experimental setup

4.2.1. Model architecture

The methodology and source codes used in this research are part of the Assessor module from the project “Strong AI for Industry”, which focuses on developing and testing a library of robust AI algorithms for the generative design of physical and digital objects.

The Assessor module includes AI-based classifiers designed to determine an object's state based on its descriptive attributes. This involves attribute extraction, normalization, and state assessment. These multi-label classifiers are also capable of handling security-related classification tasks. All models are implemented using TensorFlow [30] and TensorFlow.Keras, as part of the open-source Foressment AI library. The source code used can be found at https://github.com/aimclub/foressment_lib (Assessor AI).

We aim to determine the optimal parameters for the blocks and layer units in our dynamically customized models. To achieve this, we use Optuna [33], a hyperparameter optimization framework, to fine-tune each model. The key hyperparameters include:

- **Blocks:** specifies the number of functional blocks in the model, which can range from 1 to 3 and is applicable to Conv, Xception, and Residual.
- **Units:** specifies the number of units in each CNN layer, ranging from 64 to 256 in increments of 32, and is applicable to all models.
- **Epoch:** The parameter is set to 5 for the test run.

The selection of hyperparameters is guided by the dual objectives of achieving sufficient model complexity to extract patterns necessary for robust multi-label classification while maintaining practicality in an IIoT environment. Overly complex models, while potentially accurate, are often impractical due to increased computational demands—a critical constraint in industrial applications. We limit the number of functional blocks to a range of 1 to 3, providing adequate abstraction for processing 1D data while avoiding excessive depth that may lead to overfitting. Similarly, the number of units in each CNN layer is set between 64 and 256, as a lower unit count (e.g., 32) risks underfitting, while excessively high counts can cause overfitting. Overly complex models tend to over-interpret feature–target relationships and memorize outliers, leading to overfitting [34]. Although residual learning can mitigate overfitting in very deep models [35], our focus on 1D data makes the large unit counts typically used in image classification models (e.g., 384 or 512 units) unnecessary. In summary, these hyperparameters were strategically chosen to balance model complexity and practicality, ensuring the CNN models are efficient, effective, and suitable for deployment in IIoT settings.

Using these parameters, the function performs iterative testing to optimize the objective, which is the model's validation accuracy. It tests various combinations of units and blocks, trains the model on the training set for 5 epochs, evaluates performance on the validation set, and returns the validation accuracy. By maximizing this objective, Optuna identifies the optimal values for the “blocks” and “units” hyperparameters, achieving the highest validation accuracy after the quick 5-epoch training. With the best configuration determined, the model is constructed using these parameters and fully trained. All models are compiled with the Adam optimizer (learning rate: 0.0001) and categorical cross-entropy loss.

4.2.2. Data preparation and training settings


The Edge IIoT dataset was split with 75% of the data allocated for training and 25% for testing. Stratification based on labels was employed to ensure a balanced class representation across these splits, thus mitigating potential bias and enhancing model robustness.

CICIoT2023 dataset, given its substantial volume with 169 separate data files, was divided into two distinct groups: 160 files for training

Table 3
CICIoT2023 dataset.

Attack type	Count	Share, %	Attack type	Count	Share, %
DDoS-ICMP_Flood	7 200 504	15.423	DDoS-UDP_Fragmentation	286 925	0.615
DDoS-UDP_Flood	5 412 287	11.593	DNS_Spoofing	178 911	0.383
DDoS-TCP_Flood	4 497 667	9.634	Recon-HostDiscovery	134 378	0.288
DDoS-PSHACK_Flood	4 094 755	8.771	Recon-OSScan	98 259	0.21
DDoS-SYN_Flood	4 059 190	8.695	Recon-PortScan	82 284	0.176
DDoS-RSTFINFlood	4 045 285	8.665	DoS-HTTP_Flood	71 864	0.154
DDoS-SynonymousIP_Flood	3 598 138	7.707	VulnerabilityScan	37 382	0.08
DoS-UDP_Flood	3 318 595	7.108	DDoS-HTTP_Flood	28 790	0.062
DoS-TCP_Flood	2 671 445	5.722	DDoS-SlowLoris	23 426	0.05
DoS-SYN_Flood	2 028 834	4.346	DictionaryBruteForce	13 064	0.028
BenignTraffic	1 098 195	2.352	SqlInjection	5245	0.011
Mirai-greeth_flood	991 866	2.125	BrowserHijacking	5859	0.013
Mirai-udpplain	890 576	1.908	CommandInjection	5409	0.012
Mirai-greip_flood	751 682	1.61	Backdoor_Malware	3218	0.007
DDoS-ICMP_Fragmentation	452 489	0.969	XSS	3846	0.008
MITM-ArpSpoofing	307 593	0.659	Uploading_Attack	1252	0.003
DDoS-ACK_Fragmentation	285 104	0.611	Recon-PingSweep	2262	0.005

Table 4
Evaluation metrics.

Name	Description	Equation
Recall	Predicted positive cases over all positive cases; measures model's ability in detecting positive cases.	$\frac{TP}{TP+FN}$
Precision	True positive cases over predicted positive case; measures model's ability in assigning positive cases to the positive class.	$\frac{TP}{TP+FP}$
AUC	Area under the ROC-curve, where ROC is a chart that visualizes the trade-off between TPR and FPR .	
Accuracy	Measures overall predicting ability.	$\frac{TP+TN}{TP+FP+TN+FN}$
F1	Another statistical measure of the predicting ability — harmonic average.	$\frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$

TP – true positive, TN – true negative, FP – false positive, and FN – false negative.

and the remaining 9 for testing. To effectively manage the extensive data during training, the files were further segmented into 16 batches. Each batch adheres to the same 75%/25% split strategy as implemented in the Edge IIoT dataset, ensuring consistent training and testing conditions across all the batches. Consequently, in each subset, the data is representative, and the testing conditions are uniformly applied.

For Edge-IIoT, all models undergo 20 epochs, while for CICIoT2023, where the associated tasks present greater challenges due to the number of classes and imbalanced data, all models are trained for 20 epochs in each batch with parameters tuning during the continuous training.

4.2.3. Evaluation metrics

We used five major evaluation metrics as illustrated in Table 4.

While *accuracy* can provide an overall measure of performance, it is more crucial to maximize attack detection ability and minimize false alarms in intrusion detection tasks. Among the selected metrics, *recall* reflects the detection ability of the model, *precision* indicates how sensitive the model is at identifying attacks from benign behavior, and a low *precision* rate indicates a high number of false alarms. Hence, it is essential to look into *recall* and *precision* of minorities for better performance evaluation. Additionally, both the AUC and F1 provide a more comprehensive evaluation of model performance, particularly for imbalanced classification tasks.

4.3. Results of experiments

Table 5 presents performance metrics of all the models with their optimized best structures, determined using Optuna. The major parameters Optuna tuned are “Block” and “Unit”, representing the number of

functional blocks implemented and the layers’ output units respectfully. The evaluation matrix includes total training time for 20 epochs, categorical cross-entropy loss, accuracy, recall, precision, and AUC score. In order to assess the model’s capability, we compared our models’ evaluation metrics against two recent research studies that also utilized the IIoT dataset as a benchmark: ANN from [11], and deep CNN and CNN+LSTM from [12].

All four models in this study demonstrate remarkable performance, with both accuracy and AUC exceeding 0.99. With only 2 blocks and 128 units, Resnet demonstrates a remarkable balance between model complexity and computational efficiency. It attains the highest levels of accuracy (0.9980) and F1 (1.0000) among the evaluated models, alongside a competitive AUC of 0.9977. Notably, its training duration of 898.11 s is substantially shorter than those required by Xception and CNN-GRU, underscoring its superior computational efficiency. CNN-GRU and Xception presents comparable performance metrics, however, the increased complexity and higher computational demand make them less suitable for IIoT environments where resources and processing time are constrained. In contrast, although the Conv model demonstrates a high recall, its precision and F1 fall short when compared to Resnet.

While all of our models outperform the benchmark in accuracy, only ResNet and Xception surpass the ANN from [11] across all metrics. Simpler architectures like CNN3D and CNN+LSTM lag significantly in most metrics compared to more advanced models, highlighting the importance of sufficient complexity for effectively processing IIoT data. The CNN+LSTM model from [12], consisting of only one LSTM layer, reflects a reasonable design choice given the computational intensity of LSTMs; however, our results show that this simplification negatively impacts performance. IIoT environments typically generate large volumes of data requiring real-time or near-real-time processing. ResNet, with its skip connections that address the vanishing gradient problem, excels in learning from complex IIoT data streams. Experimental results show that ResNet achieves high precision and recall with fewer blocks and smaller layer units, demonstrating both its efficiency and effectiveness in addressing IIoT data challenges.

Table 6 presents performance metrics of all the models with their optimized structures on the CICIoT2023 dataset. Similarly, the major parameters Optuna tuned are “block” and “unit” for model structure manipulation. The model parameters are determined using Optuna for the first batch of loaded data. In order to further evaluate performance, we compare our results with two recent research papers that also utilize the same dataset: Gradient Boost (GB) and MLP from [13]; DNN with the best 34-class classification results from [14].

Based on the evaluation metrics, both the Conv model and CNN-GRU model demonstrate excellent overall performance. The CNN-GRU model achieves the highest accuracy, precision, and AUC, along with the second-highest recall. In contrast, the Conv model excels in attack detection, achieving the highest recall (0.9986) and F1 score (0.9609),

Table 5
Evaluation metrics for Edge IIoT.

Model	Block	Unit	Time	Accuracy	Recall	Precision	F1	AUC
Conv	3	256	1257.49	0.9979	1.0000	0.9745	0.9806	0.9871
CNN-GRU	–	224	2677.84	0.9975	0.9969	0.9985	1.0000	0.9977
Resnet	2	128	898.11	0.9980	0.9971	0.9986	0.9999	0.9978
Xception	3	128	3771.58	0.9975	0.9966	0.9977	0.9999	0.9972
ANN [11]				0.9950	0.9800 Attack 0.9980 Normal	0.9920 Attack 0.9950 Normal	0.9860 Attack 0.9965 Normal	
CNN3D [12]				0.9107	0.8876	0.8842	0.9197	
CNN+LSTM [12]				0.9510	0.8715	0.8925	0.8813	

Table 6
Evaluation metrics for CIIoT2023.

Model	Block	Unit	Time	Accuracy	Recall	Precision	F1	AUC
Conv	3	256	791.945	0.9247	0.9986	0.9260	0.9609	0.9873
CNN-GRU	–	224	2558.38	0.9601	0.9600	0.9602	0.9601	0.9998
Resnet	2	128	848.26	0.9260	0.9235	0.9313	0.9274	0.9980
Xception	3	128	2333.03	0.9406	0.9381	0.9463	0.9422	0.9992
GB [13]				0.9500	0.9500 Attack 0.9400 Normal	0.9400 Attack 0.9500 Normal	0.9400 Attack 0.9500 Normal	
MLP [13]				0.8600	0.8600 Attack 0.8600 Normal	0.8700 Attack 0.8400 Normal	0.8700 Attack 0.8500 Normal	
DNN [14]				0.9524	0.8153	0.7396	0.7594	

while maintaining balanced performance. Both models outperform the two balanced-performance models referenced in [13]. With a training duration of 2558.38 s, CNN-GRU is significantly slower than alternatives like ResNet and Conv, which could be a limitation when rapid deployment is critical. Among the models, Conv is the fastest, completing training in just 791.945 s and achieving the highest recall; however, it falls short on other metrics compared to CNN-GRU. Despite its slower speed, the CNN-GRU model is the most suitable choice in this experiment, delivering the best balance across key metrics: accuracy, recall, precision, F1, and AUC. Meanwhile, the Conv model, with its highest recall and fastest training time, is a viable alternative when speed and detection are prioritized over comprehensive performance. Additionally, the Conv model's superior detection rate may indicate better performance on minority classes, warranting further investigation.

It is observed that the DNN model [14] exhibits extremely high accuracy but lower precision, recall, and F1 compared to our model. One possible reason is that the DNN model is more likely to favor the majorities. Accuracy measures how often the model correctly predicts class labels overall, but it does not differentiate between false positives and false negatives. On the other hand, the F1 balances both precision and recall into a single metric, only becoming high if both precision and recall are individually high. Therefore, the F1 should be considered a more important evaluation metric for imbalanced multi-label classification. All baseline models derived from existing research demonstrate relative efficiency when contrasted with our CNN-based architectures. DL models constructed solely with DNN layers has rapid computational speeds and minimal resource consumption, however, it is evident from the comparative results that there is a discernible compromise in performance.

In this case, models with higher accuracy but lower precision, recall, and F1 are likely to predict more false positives and false negatives on validation data, even if they exhibit high overall accuracy during training. This reduces their ability to precisely classify positive and negative cases. Ideally, we are looking for models that have a balance between accuracy and precision/recall to ensure accurate performance on real, unseen network traffic data. Therefore, our models remain highly competitive, especially considering that all models have recall, precision, and F1s above 0.9.

Another significant observation is the structure optimization of deep CNN. When considering the model architecture, it is evident that Conv employs a more intricate structure (unit of 256, block of 3) to achieve the best results in both experiments. However, it did not

surpass the performance of the other models and exhibited the lowest precision rate. Therefore, while increasing the depth and number of convolutional units in Conv can yield advantages such as improved feature learning and enhanced generalization, it is crucial to optimize the model structure instead of solely increasing its complexity. Utilizing hybrid structures like CNN-GRU is also feasible according to the experiment result, despite that GRU only runs on GPU.

5. Discussion

Firstly, we analyze the models' performance on minor classes. The primary objective of this study is to analyze the detection performance on individual classes, with a particular emphasis on minority classes. Table 7 displays the classification report on Edge IIoT for all models. All rare attacks with class shares below 1% are highlighted, specifically, Fingerprinting, MITM (Man-in-the-Middle), Normal, and Password.

As the model demonstrating the best overall performance, Resnet achieves detection rates exceeding 0.95 for the majority of network traffic – 14 out of 15 attack types. Additionally, it maintains high F1 above 0.9 for most classes, indicative of a balanced precision and recall; an exception is noted in the Fingerprinting class, attributed to its comparatively low recall rate of 0.75. Notably, given that this attack represents a minority case with a share of only 0.05%, and considering Resnet's superior performance over the other three models, its performance can still be regarded as satisfactory.

From the perspective of minority classes, all models demonstrate acceptable performance; even in the worst-case, namely Fingerprint, the lowest detection rate of 0.75 remains acceptable. For MITM, despite its minimal share of 0.02%, 3 out of 4 models exhibit exceptional precision, recall, and an F1 of 1; only Conv has an excellent detection rate of 0.99. For ransomware, although Conv has an exceptional detection rate, CNN-GRU excels with the highest F1, indicating the most balanced performance for this class. For XSS, all models have a detection rate of 1 and balanced performance.

In summary, despite the relatively low representation of the 4 minority classes in the dataset, the overall performance of the models in detecting these classes remains commendable. It is important to note, that the selected models are designed and trained to exhibit robust performance in detecting minority classes, demonstrating their ability to generalize effectively across various classes, regardless of their prevalence in the dataset. Meanwhile, it is also possible that the samples presented in the dataset are representative of their network

Table 7
Classification report for Edge IIoT.

Traffic	Share %	Conv			CNN-GRU			Resnet			Xception		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Backdoor	1.12	1.00	0.97	0.98	1.00	0.97	0.98	1.00	0.97	0.98	1.00	0.97	0.98
DDoS_HTTP	2.25	0.99	1.00	0.99	0.99	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99
DDoS_ICMP	5.25	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
DDoS_TCP	2.26	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
DDoS_UDP	5.48	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Fingerprinting	0.05	0.81	0.76	0.76	0.63	0.94	0.82	0.94	0.75	0.84	0.94	0.75	0.84
MITM	0.02	0.96	0.99	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Normal	72.83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Password	2.26	1.00	0.98	0.99	0.99	0.97	0.98	1.00	0.96	0.98	0.96	0.98	0.97
Port_Scanning	1.02	0.92	1.00	0.95	0.92	1.00	0.95	0.93	0.98	0.96	0.94	0.98	0.96
Ransomware	0.49	1.00	0.89	0.94	1.00	0.89	0.94	0.90	0.95	0.92	0.93	0.93	0.93
SQL_injection	2.31	0.99	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.99	0.99	0.99	0.99
Uploading	1.70	0.98	1.00	0.97	0.99	0.98	0.98	0.97	1.00	0.99	0.98	1.00	0.99
Vulnerability_scanner	2.26	1.00	0.99	0.99	1.00	0.99	0.99	0.99	0.99	0.99	0.99	1.00	0.99
XSS	0.72	0.99	1.00	0.99	0.99	1.00	0.99	0.98	1.00	0.99	0.99	1.00	0.99

Table 8
Classification report for CIIoT2023.

Traffic	Share %	Conv			CNN-GRU			Resnet			Xception		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Backdoor_Malware	0.007	0.7	0.16	0.26	0.46	0.50	0.48	0.33	0.39	0.36	0.11	0.30	0.16
Benign_Traffic	2.352	0.83	0.97	0.89	0.99	1.00	0.99	0.80	0.96	0.87	0.78	0.97	0.86
Browser_Hijacking	0.013	0.70	0.20	0.31	1.00	0.83	0.91	0.80	0.07	0.13	0.67	0.06	0.10
Command_Injection	0.012	0.77	0.27	0.39	1.00	0.93	0.96	0.80	0.21	0.33	0.92	0.16	0.27
DDoS_ACK_Fragmentation	0.611	1.00	0.99	0.99	1.00	1.00	1.00	0.97	0.99	0.98	0.97	0.99	0.98
DDoS_HTTP_Flood	0.062	0.61	0.86	0.71	0.95	0.94	0.95	0.75	0.72	0.74	0.75	0.75	0.75
DDoS_ICMP_Flood	15.423	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
DDoS_ICMP_Fragmentation	0.969	1.00	0.99	0.99	0.97	0.99	0.98	0.97	0.99	0.98	0.96	0.99	0.97
DDoS_PSHACK_Flood	8.771	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
DDoS_RSTFIN_Flood	8.665	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
DDoS_SYN_Flood	8.695	0.95	0.89	0.92	0.98	0.92	0.95	0.99	0.87	0.93	0.97	0.97	0.97
DDoS_SlowLoris	0.050	0.85	0.95	0.90	0.99	0.91	0.95	0.67	0.81	0.73	0.60	0.82	0.69
DDoS_SynonymousIP_Flood	7.707	0.98	0.28	0.44	1.00	1.00	1.00	0.91	0.99	0.95	1.00	0.99	0.99
DDoS_TCP_Flood	9.634	1.00	0.94	0.97	0.99	0.99	0.99	0.97	0.98	0.97	1.00	0.90	0.95
DDoS_UDP_Flood	11.593	0.89	0.76	0.82	0.98	0.72	0.83	0.75	0.98	0.85	0.87	0.82	0.84
DDoS_UDP_Fragmentation	0.615	1.00	0.99	0.99	1.00	1.00	1.00	0.98	0.98	0.98	0.99	0.98	0.98
DNS_Spoofing	0.383	0.57	0.56	0.56	0.90	0.89	0.89	0.53	0.38	0.45	0.52	0.40	0.45
Dictionary_BruteForce	0.028	0.60	0.30	0.40	1.00	1.00	1.00	0.83	0.12	0.20	1.00	0.14	0.25
DoS_HTTP_Flood	0.154	0.91	0.94	0.93	0.84	0.89	0.87	0.86	0.87	0.87	0.86	0.88	0.87
DoS_SYN_Flood	4.346	0.85	0.94	0.89	0.85	0.96	0.90	0.87	0.93	0.90	0.88	0.93	0.90
DoS_TCP_Flood	5.722	0.89	0.95	0.92	0.97	0.98	0.98	0.97	0.94	0.95	0.93	0.99	0.96
DoS_UDP_Flood	7.108	0.66	0.99	0.79	0.97	0.71	0.82	0.94	0.47	0.63	0.73	0.79	0.76
MITM_Arp_Spoofing	0.659	0.84	0.61	0.71	0.98	0.97	0.97	0.80	0.61	0.70	0.81	0.61	0.70
Mirai_Greeth_Flood	2.125	0.98	0.99	0.98	0.99	1.00	0.99	0.91	0.98	0.95	0.97	0.98	0.98
Mirai_Greip_Flood	1.610	0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.96	0.97	0.95	0.96
Mirai_UDP_Plain	1.908	1.00	1.00	1.00	0.99	1.00	0.99	0.99	1.00	0.99	0.99	1.00	1.00
Recon_Host_Discovery	0.288	0.70	0.77	0.73	0.63	0.73	0.68	0.61	0.72	0.66	0.67	0.66	0.67
Recon_OS_Scan	0.210	0.58	0.37	0.45	0.49	0.14	0.21	0.49	0.17	0.25	0.47	0.19	0.27
Recon_Ping_Sweep	0.005	0.20	0.50	0.29	0.32	0.8	0.46	0.60	0.04	0.07	0.44	0.21	0.29
Recon_Port_Scan	0.176	0.29	0.47	0.36	0.47	0.33	0.39	0.46	0.24	0.31	0.43	0.30	0.36
SQL_Injection	0.011	0.67	0.14	0.23	0.44	0.08	0.13	0.31	0.09	0.14	0.29	0.08	0.12
Uploading_Attack	0.003	0.40	0.50	0.44	0.10	0.66	0.17	0.11	0.05	0.06875	0.18	0.04	0.06
Vulnerability_Scan	0.080	0.91	0.92	0.91	0.79	0.90	0.84	0.69	0.90	0.78	0.68	0.90	0.77
XSS	0.008	0.03	0.12	0.05	0.17	0.50	0.25	0.10	0.40	0.16	1.00	0.03	0.07

traffic types, which assists the models in capturing the underlying patterns and characteristics of the minority classes.

Table 8 presents classification results on CIIoT2023. As depicted by the Table, 20 out of 34 classes presented by traffic share with a percentage below 1%. In extreme cases like Uploading_Attack, the share is a mere 0.003%. The inadequacy of training examples hampers the effective learning of all models for these minority classes, leading to varying performance levels. For instance, in the case of the Backdoor_Malware with a share of 0.007%, the Conv model exhibits the highest precision, indicating high sensitivity. Conversely, the CNN-GRU model can identify 0.59 of actual cases, demonstrating balanced performance as indicated by its highest F1 (0.48). For Browser_Hijacking and Command_Injection, the CNN-GRU model exhibits excellent performance, with the highest recall and F1, and precision equal to 1. However,

for Uploading_Attack and XSS, the model demonstrates strong ability to detect these classes but falls short on precision. The Conv model has superior performance for DoS_HTTP_Flood, Recon_Host_Discovery, and Vulnerability_Scan, with balanced performance as shown by the evaluation metrics. From minority classes' perspective, it is evident that precision has been prioritized over recall in many cases, leading to reduced effectiveness. For instance, for classes like Recon_Ping_Sweep, Uploading_Attack and XSS, although all models can detect some true positive cases, the associated precision scores are relatively low if comparing with major classes.

Although all models demonstrate flawless detection and classification of minority classes in Edge IIoT, they encounter difficulties with certain classes in CIIoT2023. While the former dataset exhibits a relatively moderate imbalance distribution, enabling sufficient learning

Table 9
The effect of resampling techniques.

Methods	ResNet (Edge IIoT)				CNN-GRU (CICIoT2023)			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Nosampling	0.963	0.971	0.960	0.965	0.933	0.940	0.929	0.940
Undersampling	0.880	0.701	0.999	0.824	0.906	0.873	0.969	0.873
Oversampling	0.995	0.993	0.996	0.994	0.957	0.959	0.959	0.959
Over+under	0.998	0.999	0.997	1.000	0.960	0.960	0.960	0.960

from each class during training, the class volumes and imbalance in CICIoT2023 are significant. In the latter case, the models may struggle to learn effective patterns for these minority classes due to insufficient training examples. Furthermore, with an increased number of classes, there is likely a rise in feature variability and potential noise, which can degrade model performance on minority classes. Additionally, we observed a requirement for enhanced model complexity and dynamic structure in CICIoT2023. Conv is the most complex model applied, and it demonstrates relatively superior performance on rare attacks. As the sole model featuring a hybrid architecture, CNN-GRU also displays comparatively favorable performance.

To evaluate the effects of undersampling and oversampling methods, we conducted an ablation study using the best-performing models: ResNet for the Edge-IIoT dataset and CNN-GRU for the CICIoT2023 dataset. Without resampling, both models perform relatively well overall but struggle with class imbalance. Applying undersampling results in a significant drop in accuracy for both models, with ResNet's accuracy reduced to 88.0% and CNN-GRU's to 90.6%. However, recall improves significantly, particularly for ResNet, which achieves an exceptional recall of 99.9%. This suggests that undersampling forces the model to focus on minority class patterns but at the expense of precision. In contrast, oversampling produces substantial improvements across all metrics, while combining oversampling and undersampling achieves the highest performance for both datasets.

The results indicate that while oversampling alone achieves performance comparable to the hybrid approach, undersampling on its own is less effective due to the loss of valuable majority class information. However, compared to oversampling alone, the hybrid approach can reduce the training cost of the main model. This is because the adaptive undersampling method significantly reduces the number of majority class samples while maintaining sufficient diversity within the majority class to preserve model performance (see Table 9).

Lastly, we analyze how the choice of hyperparameters and different model structures affect performance. Table 10 presents the experimental results from Optuna on dataset Edge-IIoT, where the algorithm

selects random values from a specified range of hyperparameters and determines the best model configuration based on accuracy.

For Conv block, the best accuracy (0.9973) is achieved with blocks = 3 and units = 256; similarly, for CNN-GRU, the best accuracy (0.9969) is observed with units = 224. For these two models, a clear pattern emerges, demonstrating that the more complex the models, the better their performance, as they likely capture more intricate patterns in the data. However, with architectures like Xception and ResNet, the trend is different. For ResNet, the best performance (0.9965) is achieved with blocks = 2 and units = 128, while for Xception, the highest accuracy (0.9962) is obtained with blocks = 3 and units = 128. These results suggest that overly complex configurations may not necessarily enhance performance for these models. Moderate complexity can strike the optimal balance between model capacity and generalization, preventing potential overfitting or inefficiencies during training. The diverse observation highlights the importance of tailoring hyperparameters to the specific model architecture to achieve optimal performance.

Although our models perform well overall, there is certainly space for improvement. For instance, tackling the complex task of CICIoT2023 with 34 imbalanced classes requires additional techniques, like oversampling the minority cases and optimizing loss weights. The proposed GAN-based oversampler has demonstrated the ability to generate realistic synthetic samples and enhance performance on minority classes. However, this approach may not be practical in scenarios where severe resource constraints are present. In addition to SMOTE and the KNN-based adaptive oversampling method mentioned in the related work, several notable oversampling techniques are available: a selective oversampling technique utilizes outlier detection to identify representative samples from minor classes for synthetic oversampling [36]; generating synthetic samples based on the degree of classification contribution, ensuring the spatial distribution of original samples on class boundaries, and avoiding oversampling from noisy points [37]; taking into account both the absolute imbalance and the complexity of dataset classification [38]. Specifically, assigning higher loss weights to minority instances can further enhance models performance. For instance, to use a dynamically weighted loss function based on the class frequency and predicted probability for DL model compiling [39].

Another direction we plan to work on in the future is the incorporation of time-series data processing and classification techniques, which are crucial for certain such IIoT tasks as sequential attack analysis and time-series-based anomaly detection. One approach involves utilizing trend feature extraction. A representative study proposed the “trend-word” method [40], wherein the trend of a segment is defined by its

Table 10
Experiment results from optuna trails - Edge IIoT.

CNN			CNN-GRU		Xception			ResNet		
Blocks	Units	Accuracy	Units	Accuracy	Blocks	Units	Accuracy	Blocks	Units	Accuracy
1	64	0.9877	64	0.9965	1.00	128	0.9952	1.0000	64	0.9927
1	160	0.9927	64	0.9967	1.0000	128	0.9937	1.0000	64	0.9931
1	160	0.9950	64	0.9966	1.0000	160	0.9939	1.0000	96	0.9922
2	96	0.9969	128	0.9961	1.0000	192	0.9917	1.0000	224	0.9922
2	96	0.9970	160	0.9964	1.0000	224	0.9939	2	64	0.9941
2	96	0.9943	192	0.9965	1.0000	224	0.9944	2	128	0.9956
2	192	0.9955	224	0.9969	1.0000	224	0.9928	2	160	0.9948
2	192	0.9969			2	96	0.9959	2	160	0.9942
2	224	0.9971			2	96	0.9960	2	192	0.9955
3	128	0.9966			2	128	0.9965	2	192	0.9939
3	192	0.9971			2	128	0.9950	3	64	0.9949
3	192	0.9970			2	160	0.9942	3	96	0.9957
3	192	0.9960			2	224	0.9949	3	128	0.9953
3	224	0.9968			2	256	0.9948	3	128	0.9956
3	224	0.9962			3	64	0.9958	3	128	0.9962
3	224	0.9969			3	64	0.9954	3	192	0.9952
3	256	0.9973			3	64	0.9937	3	224	0.9961
3	256	0.9968			3	64	0.9951	3	224	0.9939
3	256	0.9966			3	96	0.9959	3	256	0.9951
3	256	0.9962			3	96	0.9958	3	256	0.9945

starting and ending points, and the first-order difference sequence is employed to symbolize the trend value. Another particularly valuable approach for anomaly detection involves earliness-aware feature selection. In a recent filter-wrapper approach [41], an iterative neural filter based on Euclidean distance estimation was employed, in conjunction with multiple MLPs as candidate models, to identify and capture time series features in the input vectors.

6. Conclusion

The complexity of IIoT network traffic classification necessitates a sophisticated and dynamic detection mechanism. In this study, we systematically compared various CNN-based model structures for IIoT intrusion detection. Comparisons with state-of-the-art approaches validate their effectiveness. Our distributed and continuous training approach ensures adaptability in real-world applications. Achieving an accuracy of 0.99 and F1 of 0.99 on the Edge-IIoT dataset demonstrates extraordinary performance, while an accuracy of 0.95 and F1 of 0.95 on CIIoT2023 showcases adaptability in challenging classification tasks.

In summary, this research contributes significantly through a systematic analysis of CNN models for IIoT security, an efficient distributed training approach, and experimental validation of high accuracy and detection capability. Our future plans include two improvements to enhance individual performance against rare attacks. These improvements involve the implementation of different oversampling techniques and a task-specific weight loss optimization algorithm.

CRedit authorship contribution statement

Huiyao Dong: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Igor Kotenko:** Writing – review & editing, Validation, Supervision, Resources, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Dmitry Levshun:** Writing – review & editing, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the Analytical Center for the Government of the Russian Federation (IGK 000000D730324P540002), agreement No. 70-2021-00141.

Data availability

We used publicly available datasets in our research. All references are provided in the manuscript.

References

- [1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, M. Gidlund, Industrial internet of things: Challenges, opportunities, and directions, *IEEE Trans. Ind. Inform.* 14 (11) (2018) 4724–4734.
- [2] D. Levshun, Y. Chevalier, I. Kotenko, A. Chechulin, Design and verification of a mobile robot based on the integrated model of cyber-physical systems, *Simul. Model. Pr. Theory* 105 (2020) 102151.
- [3] D. Levshun, A. Chechulin, I. Kotenko, Design of secure microcontroller-based systems: application to mobile robots for perimeter monitoring, *Sensors* 21 (24) (2021) 8451.
- [4] T. Gebremichael, L.P.I. Ledwaba, M.H. Eldefrawy, G.P. Hancke, N. Pereira, M. Gidlund, J. Akerberg, Security and privacy in the industrial internet of things: Current standards and future challenges, *IEEE Access* 8 (2020) 152351–152366, <http://dx.doi.org/10.1109/ACCESS.2020.3016937>.
- [5] M. Nuaimi, L. Fourati, B. Hamed, Intelligent approaches toward intrusion detection systems for industrial internet of things: A systematic comprehensive review, *J. Netw. Comput. Appl.* (ISSN: 1084-8045) 215 (2023) 103637, <http://dx.doi.org/10.1016/j.jnca.2023.103637>.
- [6] L. Mohammadpour, T.C. Ling, C.S. Liew, A. Aryanfar, A survey of CNN-based network intrusion detection, *Appl. Sci.* 12 (16) (2022) <http://dx.doi.org/10.3390/app12168162>.
- [7] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, J. Chen, DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system, *Secur. Commun. Netw.* 2020 (2020) 1–11.
- [8] H. Dong, I. Kotenko, Hybrid multi-task deep learning for improved IoT network intrusion detection: Exploring different CNN structures, in: 2024 16th International Conference on Communication Systems & Networks, COMSNETS, 2024, pp. 7–12, <http://dx.doi.org/10.1109/COMSNETS59351.2024.10426924>.
- [9] L. Wen, X. Li, L. Gao, Y. Zhang, A new convolutional neural network-based data-driven fault diagnosis method, *IEEE Trans. Ind. Electron.* 65 (7) (2018) 5990–5998, <http://dx.doi.org/10.1109/TIE.2017.2774777>.
- [10] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, L. Cui, Robust detection for network intrusion of industrial IoT based on multi-CNN fusion, *Measurement* 154 (2020) 107450.
- [11] A. Dehlaghi-Ghadim, M.H. Moghadam, A. Balador, H. Hansson, Anomaly detection dataset for industrial control systems, *IEEE Access* 11 (2023) 107982–107996, <http://dx.doi.org/10.1109/ACCESS.2023.3320928>.
- [12] S.H. Javed, M.B. Ahmad, M. Asif, W. Akram, K. Mahmood, A.K. Das, S. Shetty, APT adversarial defence mechanism for industrial IoT enabled cyber-physical system, *IEEE Access* 11 (2023) 74000–74020, <http://dx.doi.org/10.1109/ACCESS.2023.3291599>.
- [13] A.S. Jaradat, A. Nasayreh, Q. Al-Na'ameh, H. Gharaibeh, R.E. Al Mamlook, Genetic optimisation techniques for enhancing web attacks classification in machine learning, in: 2023 IEEE DASC/PiCom/CBDCom/CyberSciTech, 2023, pp. 0130–0136, <http://dx.doi.org/10.1109/DASC/PiCom/CBDCom/Cy59711.2023.10361399>.
- [14] K.R. Narayan, S. Mookherji, V. Odelu, R. Prasath, A.C. Turlapaty, A.K. Das, IIDS: Design of intelligent intrusion detection system for internet-of-things applications, 2023, [arXiv:2308.00943](https://arxiv.org/abs/2308.00943).
- [15] A. Yazdinejad, A. Dehghantanha, G. Srivastava, H. Karimipour, R.M. Parizi, Hybrid privacy preserving federated learning against irregular users in next-generation internet of things, *J. Syst. Archit.* (ISSN: 1383-7621) 148 (2024) 103088, <http://dx.doi.org/10.1016/j.jsysarc.2024.103088>.
- [16] A. Yazdinejad, A. Dehghantanha, R.M. Parizi, M. Hammoudeh, H. Karimipour, G. Srivastava, Block hunter: Federated learning for cyber threat hunting in blockchain-based IIoT networks, *IEEE Trans. Ind. Inform.* 18 (11) (2022) 8356–8366, <http://dx.doi.org/10.1109/TII.2022.3168011>.
- [17] A. Yazdinejad, A. Dehghantanha, R.M. Parizi, G. Srivastava, H. Karimipour, Secure intelligent fuzzy blockchain framework: Effective threat detection in IIoT networks, *Comput. Ind.* (ISSN: 0166-3615) 144 (2023) 103801, <http://dx.doi.org/10.1016/j.compind.2022.103801>.
- [18] T. Gaber, J.B. Awotunde, S.O. Folorunso, S.A. Ajagbe, E. Eldesouky, Industrial internet of things intrusion detection method using machine learning and optimization techniques, *Wirel. Commun. Mob. Comput.* 2023 (1) (2023) 3939895, <http://dx.doi.org/10.1155/2023/3939895>.
- [19] M. Zolanvari, M.A. Teixeira, R. Jain, Effect of imbalanced datasets on security of industrial IoT using machine learning, in: 2018 IEEE International Conference on Intelligence and Security Informatics, ISI, 2018, pp. 112–117, <http://dx.doi.org/10.1109/ISI.2018.8587389>.
- [20] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, *IEEE Trans. Syst. Man Cybern.* (3) (1972) 408–421.
- [21] I. Tomek, Two modifications of CNN, *IEEE Trans. Syst. Man Cybern.* SMC-6 (11) (1976) 769–772, <http://dx.doi.org/10.1109/TSMC.1976.4309452>.
- [22] T. Gaber, J.B. Awotunde, M. Torky, S.A. Ajagbe, M. Hammoudeh, W. Li, Metaverse-IDS: Deep learning-based intrusion detection system for metaverse-IIoT networks, *Internet Things* (ISSN: 2542-6605) 24 (2023) 100977, <http://dx.doi.org/10.1016/j.iot.2023.100977>.
- [23] H. Dong, I. Kotenko, An autoencoder-based multi-task learning for intrusion detection in IIoT networks, in: 2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology, USBREIT, 2023, pp. 1–4, <http://dx.doi.org/10.1109/USBREIT58508.2023.10158807>.
- [24] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Zomaya, R. Ranjan, A hybrid deep learning-based model for anomaly detection in cloud datacenter networks, *IEEE Trans. Netw. Serv. Manag.* (2019) <http://dx.doi.org/10.1109/TNSM.2019.2927886>, 1–1.
- [25] Enhancing IIoT network security through deep learning-powered intrusion detection system, *Internet Things* (ISSN: 2542-6605) 24 (2023) 100936, <http://dx.doi.org/10.1016/j.iot.2023.100936>.
- [26] H. Hotelling, Analysis of a complex of statistical variables into principal components, *J. Educ. Psychol.* 24 (6) (1933) 417.

- [27] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations, ICLR 2015, Computational and Biological Learning Society, 2015.
- [28] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1251–1258.
- [29] S. Targ, D. Almeida, K. Lyman, Resnet in resnet: Generalizing residual architectures, in: *Proceedings of the ICLR 2016 Workshop*, 2016.
- [30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [31] M.A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, H. Janicke, Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications: Centralized and federated learning, 2022, <http://dx.doi.org/10.21227/mbc1-1h68>.
- [32] E.C.P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, A.A. Ghorbani, CICIOT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment, *Sensors* (ISSN: 1424-8220) 23 (13) (2023) <http://dx.doi.org/10.3390/s23135941>.
- [33] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.
- [34] P. Chen, H.-H. Chen, Accelerating matrix factorization by overparameterization, in: *DeLTA*, 2020, pp. 89–97.
- [35] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 770–778, <http://dx.doi.org/10.1109/CVPR.2016.90>.
- [36] P. Gnip, L. Vokorokos, P. Drotár, Selective oversampling approach for strongly imbalanced data, *PeerJ Comput. Sci.* 7 (2021) e604.
- [37] Z. Jiang, T. Pan, C. Zhang, J. Yang, A new oversampling method based on the classification contribution degree, *Symmetry* 13 (2) (2021) 194.
- [38] D. Lee, K. Kim, An efficient method to determine sample size in oversampling based on classification complexity for imbalanced data, *Expert Syst. Appl.* 184 (2021) 115442.
- [39] K.R.M. Fernando, C.P. Tsokos, Dynamically weighted balanced loss: Class imbalanced learning and confidence calibration of deep neural networks, *IEEE Trans. Neural Netw. Learn. Syst.* 33 (7) (2022) 2940–2951, <http://dx.doi.org/10.1109/TNNLS.2020.3047335>.
- [40] B. Bai, G. Li, S. Wang, Z. Wu, W. Yan, Time series classification based on multi-feature dictionary representation and ensemble learning, *Expert Syst. Appl.* 169 (2021) 114162.
- [41] M. Kherbache, K. Amroun, D. Espes, A new wrapper feature selection model for anomaly-based intrusion detection systems, *Internat. J. Security Netw.* 17 (2) (2022) 107–123.