

ARTICLE

Zero-day attack detection in Internet of Things networks using deep transductive transfer learning

Gunupusala Satyanarayana^{1*}, Mohammad Sirajuddin², Nimmala Mangathayaru³, Emandi Sreedevi⁴, Jhansi Lakshmi Sarwani Theeparthi⁵, Pasi Ashok Kumar⁶

¹Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Nambur, Andhra Pradesh, India

²Department of Computer Science and Engineering, Kaveri University, Siddipet, Telangana, India

³Department of Information Technology, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Telangana, India

⁴Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India

⁵Department of Computer Science and Engineering, Aditya University, Surampalem, Andhra Pradesh, India

⁶Department of Computer Science and Engineering (Data Science), ACE Engineering College, Hyderabad, Telangana, India

***Corresponding author:**
Gunupusala Satyanarayana
(snarayana.5813@gmail.com)

Citation: Satyanarayana, G., Sirajuddin, M., Mangathayaru, N., Sreedevi, E., Theeparthi, J. L. S. & Kumar, P. A. (2026). Zero-day attack detection in Internet of Things networks using deep transductive transfer learning. *Int J Systematic Innovation*, 10(4): 026150048.
[https://doi.org/10.6977/IJoSI.202608_10\(4\).0007](https://doi.org/10.6977/IJoSI.202608_10(4).0007)

Received: April 6, 2026

Revised: July 11, 2026

Accepted: July 14, 2026

Published online: August 24, 2026

Copyright: © 2026 Author(s). This is an Open-Access article distributed under the terms of the Creative Commons Attribution License, permitting distribution, and reproduction in any medium, provided the original work is properly cited.

Publisher's Note: AccScience Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Abstract

Cyberattacks targeting Internet of Things (IoT) systems, particularly zero-day exploits, are escalating due to inherent vulnerabilities in IoT networks. Traditional intrusion detection systems (IDSs) use machine learning, such as deep learning (DL), to improve cyberattack detection. Nevertheless, DL-based IDSs require well-balanced datasets with abundant labeled data, which is often not available in IoT networks. In this article, we propose an efficient IDS that incorporates transfer learning (TL), knowledge transfer, and model refinement to accurately identify zero-day attacks. The TL model is based on deep convolutional neural networks adapted to 5G IoT environments with unbalanced and limited labeled datasets. The proposed framework employed three specialized datasets: the University of New South Wales Network-Based 2015 dataset (UNSW-NB15)-Basic for model training, UNSW-NB15-Test+ for evaluating zero-day attack detection, and UNSW-NB15-Test for comprehensive evaluation involving both known and zero-day attacks. The experimental results validate the effectiveness of our approach, achieving high accuracy and low false-prediction rates. In particular, the introduced TL-oriented solution outperforms other DL-based IDSs in detecting various families of known and zero-day attacks, representing a significant step forward in protecting IoT devices against cyber threats.

Keywords: Cybersecurity; Convolutional neural network; Intrusion detection systems; Internet of Things networks; Transfer learning

1. Introduction

The role of intrusion detection systems (IDSs) capable of detecting zero-day cyberattacks is vital for addressing the exponential growth of such attacks (Hindy *et al.*, 2018; Kaloudi & Li, 2021). Machine learning (ML) approaches have been widely used to develop strong IDSs (Hindy *et al.*, 2018; Khraisat *et al.*, 2019). However, current IDSs can detect known attacks with high precision but often fail to detect new or zero-day attacks because they rely on predefined patterns and signatures, which are inherently limited (Chapman, 2016). Another problem with existing IDSs is that they produce many false positives, thus reducing their efficiency and usefulness for real-world applications (Bilge & Dumitras, 2012). Therefore, many undisclosed threats still pass unnoticed, resulting in service unavailability and customer data loss due to identity theft, among other issues (Nguyen & Reddi, 2023).

A zero-day attack occurs when an exploit of a previously unknown software, firmware, or hardware vulnerability is discovered and used before a patch or detection signature is available. The vulnerability is not known to software vendors and security defenders, making it difficult for signature-based IDSs to detect such an attack. In an Internet of Things (IoT) environment, zero days are especially problematic because many devices are constrained in their computing capabilities, receive infrequent software/hardware updates, have varying architectures, and have extended deployment cycles. Post-compromise, attackers can exploit vulnerable IoT devices to access them, exfiltrate sensitive data, impact critical services, or embed them in a large botnet, making zero-day attack detection challenging, as it requires intelligent anomaly detection methods that can identify deviations from normal network behavior, rather than simply looking for known attack signatures. Bilge and Dumitras (2012) thoroughly analyzed the real-world impact of such attacks and attempted to understand how widely they are spread. Their study shows that there are more zero-day attacks than previously thought: of the 18 attacks they analyzed, 11 (61%) were new. Moreover, their findings reveal that these types of attacks can go unnoticed for a long time; on average, 10 months before detection, during which time they can compromise systems. Moreover, 62% of attacks are identified only after they have infiltrated systems (Nguyen & Reddi, 2023). Another study showed that the number of zero-day attacks in 2019 was higher than the sum of the previous three years (Metrick *et al.*, 2020). These discoveries highlight the urgent need for better models to detect attacks. One way researchers attempt to detect zero-day attacks is by looking for outliers (i.e., instances that differ from normal traffic). However, current outlier-based detection methods suffer

from two problems: they produce too many false positives and false negatives due to their low accuracy. As mentioned before, with high false-negative rates, there is still a chance that an attack can succeed, while high false-positive rates waste cybersecurity operation centers' time; only 28% of investigated intrusions are real (Hindy *et al.*, 2020). For example, Ficke *et al.* (2019) pointed out how false negatives limit IDS development by lowering its effectiveness.

Machine learning and deep learning (DL) methods have been widely used in IDSs, which are considered the main focus of current research on IoT security. Nevertheless, it was commonly found (Kilincer *et al.*, 2021) that ML had weak feature engineering at the outset, which might result in low detection rates (Fadlullah *et al.*, 2017). Additionally, various threats and intrusions could not be detected by ML-based solutions when attacks were unforeseen or unidentifiable; hence, DL techniques were introduced. Unlike traditional approaches that rely on patterns derived from normal behavior to identify attacks, DL can detect attacks with greater precision by leveraging abnormal patterns while minimizing false alarms (Rodriguez *et al.*, 2021; Thamilarasu & Chawla, 2019). DL-based IDSs can learn complex patterns from large volumes of labeled data, enabling the training of classification models for intrusion detection.

Although DL has shown promising results in intrusion detection, it requires a large, well-balanced, and well-labeled training set for optimal performance. In the IoT setting, however, such data is scarce because gathering representative attack traffic is costly and time-consuming, and impractical for a variety of privacy, security, and deployment reasons. Moreover, the constant emergence of new attack families puts traditional DL models in the position of needing to be retrained on fresh samples, which incurs significant computational resources and delays deployment. The challenges are further compounded by the heterogeneous nature and limitations of computing resources in IoT devices, which makes the development of scalable and adaptive IoT IDSs especially difficult. To overcome this, reusing knowledge acquired in related domains has proven a useful approach, making transfer learning an effective solution when significant class imbalance and scarcity of labeled data are present (Rodriguez *et al.*, 2021).

Transfer learning leverages knowledge acquired from related source domains to improve performance on target domains by transferring learned features or representations between related tasks. This approach enables models trained on high-performing or data-rich domains to improve learning in low-performing or data-scarce domains. This can be considered a recent advancement

in the field of ML, which uses knowledge from one area to solve problems in another. Transfer learning has had remarkable success in natural language processing (Ruder *et al.*, 2021) and computer vision (Gopalakrishnan *et al.*, 2017), where model performance can be greatly improved by training image classification models on different object categories and then transferring knowledge to new, related domains. However, one important observation from these studies is that better results were achieved when some level of understanding of the target problem was gained prior to applying transfer learning techniques, i.e., rather than training models blindly on fresh datasets from scratch.

Recent examinations of transfer learning in IDSs have yielded useful results, particularly in domains with limited data, such as IoT networks, where it is used to increase the detection rate for known attacks and to identify zero-day attacks. Research on TL has shown that it can be applied to the design of new IDSs to improve detection accuracy under different situations, such as when there are few records of previous attacks; this also helps speed up the training process and reveals day-zero attack cases. In light of this, the present study extends beyond previous work, which has primarily focused on describing existing approaches, by proposing a unified and efficient transfer learning-based framework for detecting both known and zero-day attacks in IoT networks. The main goal of such research is to create an effective system that detects intrusions more accurately than others by sharing knowledge between models during their refinement stages, while using limited resources in unbalanced dataset environments commonly found in the internet-connected devices sector. Hence, it deepens our understanding of cybersecurity threats to these gadgets, which often lag in updates due to their limited memory capacities.

Deep learning and transfer learning are somewhat related but different concepts. DL involves multilayered neural networks that are trained to learn hierarchical representations from the training set, and typically tries to optimize the entire network for the target task automatically (LeCun *et al.*, 2015). Transfer learning, on the other hand, reuses knowledge gained from a source task or source domain to enhance learning in a related target task or target domain. Thus, transfer learning is not a new neural network architecture, but rather a learning strategy that can be applied to deep neural networks (DNNs). Deep transfer learning involves training a model on a source dataset and then using the learned parameters or feature representations for the target task. The higher layers can be fine-tuned or substituted to suit the classes in the target domain (Yosinski *et al.*, 2014), and the early layers can be kept as general feature extractors. This can

reduce the number of targets needed for training and the computational resources required, compared with training a new deep model with randomly initialized parameters.

To do this, we examined two datasets in the IDS space, comprising normal and cyberattack traffic flows in the IoT domain. The BoT-IoT dataset was selected as our source domain because it covers a wide range of network traffic in IoT devices (Koroniotis *et al.*, 2019). On the other hand, the University of New South Wales Network-Based 2015 dataset (UNSW-NB15) was selected as the target domain because it comprises labeled data on contemporary cyberattacks involving IoT networks (Moustafa & Slay, 2015).

The main contributions of this paper are as follows: (i) we propose a new framework for detecting known and zero-day attacks within IoT networks based on transfer learning principles and network fine-tuning; and (ii) we suggest creating three specialized datasets to train and evaluate our framework:

- (i) UNSW-NB15-Basic has normal traffic along with four types of known attacks.
- (ii) UNSW-NB15-Test+ contains normal traffic plus five kinds of zero-days
- (iii) UNSW-NB15-Test includes normal traffic with nine different attacks (four knowns, five zeros).

The remaining parts of this paper are organized as follows. Related work on transfer learning-based solutions for cyberattack detection in IoT networks is discussed in **Section 2**. **Section 3** provides the background used in our work. The proposed transfer learning-based intrusion detection framework is presented in **Section 4**, while its usage within IoT networks and performance evaluation results are described in **Section 5**. Finally, the study's conclusions and future directions for research are presented in **Section 6**.

2. Related work

An IDS is defined as “a system or software that monitors a network or systems for malicious activity.” IDSs typically fall into two categories: network IDSs and host IDSs. Network IDS monitors network and communication activities, while host IDS monitors internal operations and log files (Hodo *et al.*, 2017). IDSs can be further classified by their detection techniques. Signature-based IDS relies on known attack signatures, while anomaly-based IDS relies on pattern recognition (Hamed *et al.*, 2018). Anomaly-based IDSs demonstrate better performance than signature-based ones when dealing with complex and unknown attacks.

Based on research over the past decade, anomaly-based IDSs have shown improvements in robustness. In the beginning, statistical methods were employed primarily to detect cyber anomalies and attacks. However, as cyberattacks became more sophisticated, these statistical techniques could no longer handle their complexity. Therefore, ML was needed due to its success in different domains such as image and video processing, among others, including natural language processing, which led to DL.

Having realized the potential of ML/DL in cybersecurity, most researchers quickly adopted it for various cyber applications. According to Nguyen & Reddi (2023), ML has many advantages over traditional rule sets, including “robust resistance” to attacks, which they argue is important for security purposes. This shift toward ML approaches represents an essential step forward within IDSs, as it allows them to better adapt to a changing cyber threat landscape (Hindy *et al.*, 2018).

According to new research, most IDSs have been based on ML in the last 10 years (Hindy *et al.*, 2018). In this IDS domain, artificial neural networks, support vector machines (SVMs), and k-means are the most common algorithms. Buczak and Guven (2016) have also examined trends and complexities related to ML methods used alongside DL techniques for IDS. It is worth noting that most people now use DL because it can handle complex patterns when analyzing network traffic.

The use of encrypted traffic has become widespread, thus making it difficult for an intruder detection system to operate. Therefore, Aceto *et al.* (2020) stated that IDS should be able to handle encrypted traffic, since traditional packet- and port-based methods were effective against only 13% of packets in early 2019.

The first work in transfer learning for intrusion detection, according to Wu *et al.* (2019), used convolutional neural network (CNN) models in a two-stage learning process. The system first learns from a base dataset, such as UNSW-NB15, and then transfers this knowledge to the target dataset, which is the Network Security Laboratory–Knowledge Discovery in Databases (NSL-KDD) dataset. Two concatenated CNNs were used in this approach and evaluated on the KDDTest-21 dataset, which includes zero-day attacks. They achieved an accuracy of 81.94%, which is 2.86% better than the traditional approach.

Convolutional neural network models can be used for detecting novel attacks. Masum & Shahriar (2020) also presented a transfer learning approach for detecting novel intrusions, but they did so in two steps. In their method, they first applied the VGG-16 model pretrained

on the ImageNet dataset, and then trained a DNN on the extracted features. DNN consists of an input layer, two hidden layers with 64 and 8 nodes, respectively, and an output layer. They achieved an accuracy of 70.97% in detecting novel intrusions (KDDTest-21), slightly lower than the previous work by Wu *et al.* (2019), where the Network Security Laboratory–Knowledge Discovery in Databases dataset was used as an evaluation set for this method, thus showing these results could be achieved even when there is a limited number of records available.

Sameera & Shashi (2019) used transfer learning techniques within IDSs, specifically targeting zero-day attack detection, particularly remote-to-local attacks; however, their system detects unlabeled remote-to-local attacks in the NSL-KDD database using labeled denial of service (DoS) attacks. This method achieved an impressive accuracy of up to 89.79%, with a false positive rate (FPR) as low as 0.15%, representing a significant improvement over previous feature-based methods, with recognition rates exceeding them by nearly 12 percentage points.

Singla *et al.* (2019) proposed a transfer learning system to detect new attacks belonging to specific families by transferring knowledge from one domain (source) to another (target) when training data is limited. Their model consisted of two DNNs with different numbers of densely connected layers and was evaluated on UNSW-NB15. The dataset was split into a source dataset, which contains various attack categories, and a target dataset, which contains a new attack type, thereby achieving an accuracy improvement of 3.2%–19.1% across different new attack types compared with DL models trained from scratch as a baseline.

Mehedi *et al.* (2021) aimed to accelerate IDS model training using deep transfer learning. They trained their model using two CNNs on two subsets of a labeled dataset for a next-generation in-vehicle network, and evaluated it against three types of attacks: flooding, fuzzing, and spoofing. They achieved an overall accuracy of 98.1%, indicating that their transfer learning-based detection model performs best for normal traffic and attack classification.

Fan *et al.* (2020) proposed a federated framework for 5G IoT environments that combines transfer learning with federated learning. Personalized intrusion detection models were developed for each IoT network using CNN-based transfer learning. The average detection accuracy was 91.93% based on evaluation results from the Canadian Institute for Cybersecurity Intrusion Detection System 2017 and custom target datasets; thus, this method enhances intrusion detection in heterogeneous IoT networks while preserving data privacy and security.

Idrissi *et al.* (2021) proposed using transfer learning to detect novel attacks in industrial IoT scenarios where there is little labeled data available by fine-tuning only certain parts (last layers) while freezing others (most layers) using a CNN and pretrained model. The source domain was the BoT-IoT dataset, which was augmented with the Telemetry and Operating Network-IoT dataset from the target domain, specific to industrial IoT, resulting in 99% accuracy for novel attack detection.

Guan *et al.* (2021) used deep transfer learning for intrusion detection in IoT environments with few labeled data, focusing on network classification. EfficientNet and Big Transfer architectures, traditionally used for image recognition, were leveraged to achieve high accuracies of 96.22% and 96.40%, respectively, on a 10%-labeled University of Science and Technology of China Traffic Flow Classification 2016 dataset.

Mehedi *et al.* (2023) proposed a deep transfer learning-based residual neural network for intrusion detection in heterogeneous IoT networks and achieved an overall accuracy of 87% across nine types of attacks and various IoT sensors.

Table 1 provides an overview of transfer learning-based intrusion detection methods applied in IoT environments; each method employs different transfer learning approaches, such as CNN-CNN, DNN-DNN, and principal component analysis-K-nearest neighbor,

among others, using various combinations of source and target datasets. The reported accuracies demonstrate that transfer learning can substantially improve intrusion detection across diverse IoT datasets and attack scenarios.

3. Background

This section provides information about the models used for our study. Section 3.1 describes the CNN model, which is based on DL, while Section 3.2 explains how we implemented the transfer learning approach. Section 3.3 explains the synthetic minority oversampling technique (SMOTE), which was employed to address class imbalance by generating synthetic samples for minority classes.

3.1. Convolutional neural networks

Image recognition and classification tasks are among the tasks to which a common DL architecture, the CNN, can be applied. This differs from traditional methods, where images are processed directly without any requirement for additional operations such as data reconstruction or feature extraction (Song *et al.*, 2020). A convolutional layer, a pooling layer, and a fully connected layer are the three main layers of a typical CNN. In convolution, the convolutional layer automatically extracts features by detecting patterns and structures in the input images. Correspondingly, pooling layers exploit local correlations to reduce complexity while preserving necessary information. Finally, the fully connected layer integrates

Table 1. Summary of transfer learning approaches used for intrusion detection in Internet of Things environments

Reference	TL approach	Source dataset	Target dataset	Accuracy
Wu <i>et al.</i> (2019)	CNN-CNN	UNSW-NB15	NSL-KDD	81.94%
Masum & Shahriar (2020)	DNN-DNN	VGG-16	NSL-KDD	70.97%
Sameera & Shashi (2019)	PCA-KNN	NSL-KDD (DoS+ Normal)	NSL-KDD (R2L+ Normal)	89.79%
Singla <i>et al.</i> (2019)	DNN-DNN	UNSW-NB15 subset	UNSW-NB15 subset	95–98%
Maz <i>et al.</i> (2025)	Ensemble classifier (TRANS-ENS)	BoT-IoT	Keylogger_detection	96.06%
Fan <i>et al.</i> (2020)	CNN-CNN	CICIDS2017	Custom	91.93%
Idrissi <i>et al.</i> (2021)	CNN-CNN	BoT-IoT	TON-IoT	99.43%
Guan <i>et al.</i> (2021)	BiT, EfficientNet	Custom	10% USTC-TFC2016	96%
Mehedi <i>et al.</i> (2023)	CNN	Custom	Custom	87%

Abbreviations: AWID: Aegean WiFi Intrusion Dataset; BiT: Big Transfer; CICIDS2017: Canadian Institute for Cybersecurity Intrusion Detection System 2017; CNN: Convolutional neural network; DNN: Deep neural network; DoS: Denial of service; IoT: Internet of Things; KNN: K-nearest neighbor; NSL-KDD: Network Security Laboratory-Knowledge Discovery in Databases; PCA: Principal component analysis; RF: Random forest; R2L: Remote-to-local; SVM: Support vector machine; TL: Transfer learning; TON-IoT: Telemetry and Operating Network-Internet of Things; UNSW-NB15: University of New South Wales Network-Based 2015 dataset; USTC-TFC2016: University of Science and Technology of China Traffic Flow Classification 2016; VGG-16: Visual Geometry Group 16.

features extracted from the previous layers to produce the final output, enabling image classification and recognition tasks. The CNN architecture, as illustrated in Figure 1, comprises three distinct layers: convolutional, pooling, and classification.

3.2. Transfer learning

Deep learning has proven useful in many applications, provided there is sufficient training data and the test data comes from the same distribution and input feature space (Zhuang *et al.*, 2020). However, when data is limited or hard to obtain, it may be necessary to create a learner capable of high performance in a target domain with few or no labeled examples. This requirement reflects the core idea of transfer learning, which leverages knowledge from a related source domain with abundant labeled examples (Zhuang *et al.*, 2020). Transfer learning helps problem-solving by leveraging previously gained experience, even when the training and testing domains, tasks, or distributions differ. For example, a model that learns to recognize cars could also learn to distinguish trucks from cars. In our case, transfer learning can be used to detect new attack families by leveraging prior knowledge from detecting existing attacks in IoT networks. As formally defined in Pan & Yang (2010), transfer learning improves learning of the target predictive function $f_T(\cdot)$ in the target domain (DT) by integrating knowledge from both the source domain (DS) and the source learning task (TS), where $DS \neq DT$ or $TS \neq TT$.

Transfer learning may be categorized into two major dichotomies (Shao *et al.*, 2019). The first is based on the availability of labeled data, resulting in three types: transductive transfer learning (also known as zero-day attack), inductive transfer learning, and unsupervised transfer learning. The second one is based on differences between feature spaces, leading to four groups: instance-based transfer learning methods; feature-based approaches; parameter-based techniques; and relational-based approaches.

Convolutional neural networks, among the most successful deep architectures, are primarily used for image recognition and classification tasks and eliminate the need for separate feature extraction followed by data reconstruction operations (Morid *et al.*, 2021). A typical CNN consists of convolutional layers, pooling layers, and fully connected layers, in which it automatically extracts features through convolution, reduces complexity via pooling, and then integrates these features to generate the output. Deep transfer learning involves transferring weights from a pretrained DL model to another model trained on a different dataset (Shao *et al.*, 2018). This has been a successful strategy in many image processing and classification tasks. Deep transfer learning uses basic features learned across tasks in the lower layers of a CNN; thus, it is applicable to all tasks, while performance is further improved through fine-tuning, which adjusts higher-order features to suit new datasets (Shao *et al.*, 2019). In our

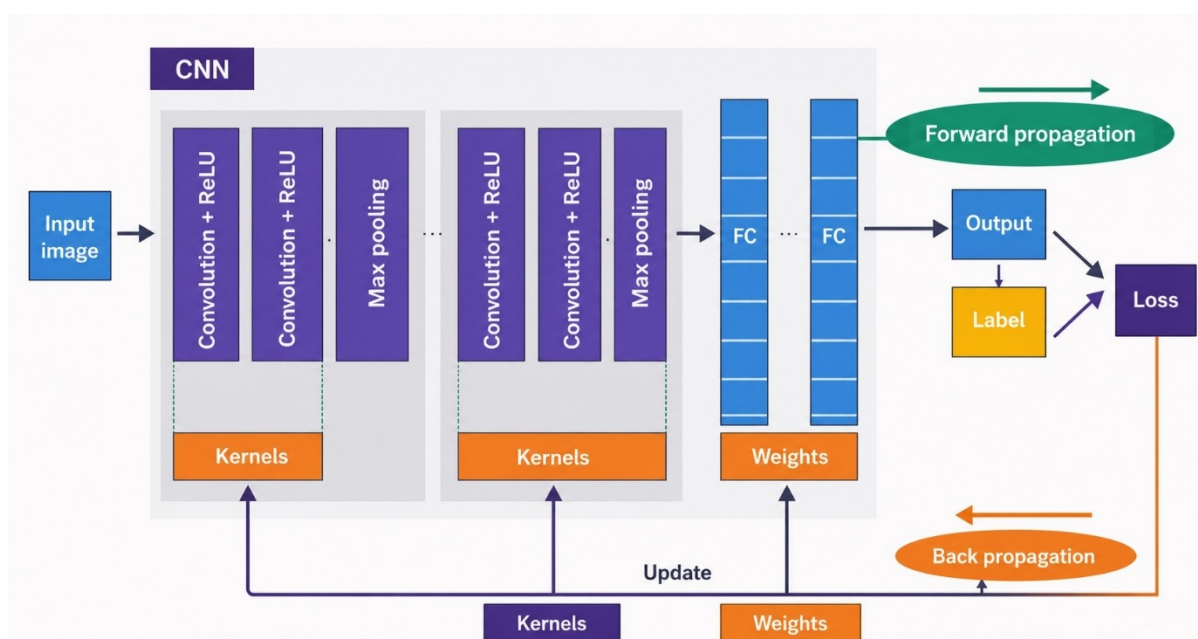


Figure 1. Convolutional neural network (CNN) architecture and the training process

proposed deep transfer learning framework, we used seven pretrained CNN architectures, namely Xception, VGG16, VGG19, Inception, InceptionResNetV2, EfficientNetB7, and EfficientNetV2L, that were initially trained on the ImageNet dataset, which is widely used as a benchmark for image processing (Morid *et al.*, 2021).

3.3. Synthetic minority oversampling technique

The SMOTE approach was employed during data preprocessing to address class imbalance in the training datasets. Rather than duplicating minority-class instances, SMOTE generates synthetic samples by interpolating between neighboring minority-class instances, thereby preserving the underlying data distribution while improving class representation. In this study, three specialized datasets were developed for training and evaluation: (i) UNSW-NB15-Basic, containing normal network traffic and four known attack categories; (ii) UNSW-NB15-Test+, comprising normal traffic and five zero-day attack categories; and (iii) UNSW-NB15-Test, including normal traffic with nine attack categories (four known and five zero-day attacks). SMOTE was applied to the UNSW-NB15-Basic training dataset to balance the distribution of known attack classes before training the transfer learning model. The balanced training data enabled the model to learn more discriminative representations of minority attack classes while reducing bias toward majority classes. The UNSW-NB15-Test+ and UNSW-NB15-Test datasets were retained in their original form for evaluation, ensuring a realistic assessment of the proposed framework's capability to detect both known and previously unseen zero-day attacks. Consequently, the use of SMOTE improved the classifier's ability to recognize rare attack patterns, reduced the misclassification of minority attack categories, and enhanced the overall generalization performance on unseen network traffic (Chawla *et al.*, 2002).

4. Proposed framework

This paper presents a deep transfer learning-based framework for intrusion detection that targets both known and novel attack families in IoT networks. The framework consists of two main phases: (i) an initial training phase conducted on the source domain, and (ii) a transfer learning phase transitioning to the target domain.

In both phases, CNNs were used as the core learning models, denoted as deep CNN (DCNN)-B and DCNN-TL, respectively, with the approach keeping the convolutional base fixed and using its output to feed the classifier. The detection process unfolds through different stages, which involve data preprocessing and transfer learning:

- (i) Stage 1: Preprocessing of the source domain dataset.
- (ii) Stage 2: Learning on the source domain (training DCNN-B with the source dataset).
- (iii) Stage 3: Preprocessing of the target domain dataset.
- (iv) Stage 4: Transfer learning to the target domain (training DCNN-TL with the target dataset).
- (v) Stage 5: Detection of attacks on the target dataset.

Figure 2 illustrates the overall structure of the proposed intrusion detection framework.

4.1. Data treatment and preprocessing

The first and third phases of a transfer learning-based IDS in IoT deal with preprocessing data from the source and target sets. These records represent all network activity, including legitimate traffic and various types of cyberattacks. Features such as flow, connection, content, and time were extracted from raw packets to create datasets for analysis. These characteristics often include nominal, integer, float, timestamp, and binary data types. Additionally, for CNN models, raw data must be converted to image format. Based on these observations, the treatment and preprocessing of IoT datasets typically consist of the following steps:

- (i) One-hot encoding transformation: Conversion of nominal fields into numeric format using the one-hot encoding method.
- (ii) Decimal conversion: Conversion of hexadecimal fields into decimal format.
- (iii) Logarithmic method: Applying a logarithmic procedure to features with values concentrated around 0.
- (iv) Standardization: Normalizing the dataset so as to prevent model overfitting and biased results.
- (v) Image transformation: Converting raw data into image format.

The specific details of the preprocessing steps in the DCNN-B algorithm are presented in Algorithm 1.

The package consisted of a complete pipeline of functions designed to preprocess data represented as a list of raw records. The output should be in a format that can be utilized by ML models, especially IDSs. The `one_hot` encoding function converted nominal fields into numeric representations using one-hot encoding. For each record in the input dataset, it produces a new encoded record and creates binary vectors for each nominal field, effectively representing categorical variables. Hexadecimal fields were converted to decimal by the `hexadecimal_to_decimal` function, ensuring uniformity and making them easier to work with numerically. The `logarithmic_transformation` function, on the other hand, applied a logarithmic transformation to fields with values heavily concentrated

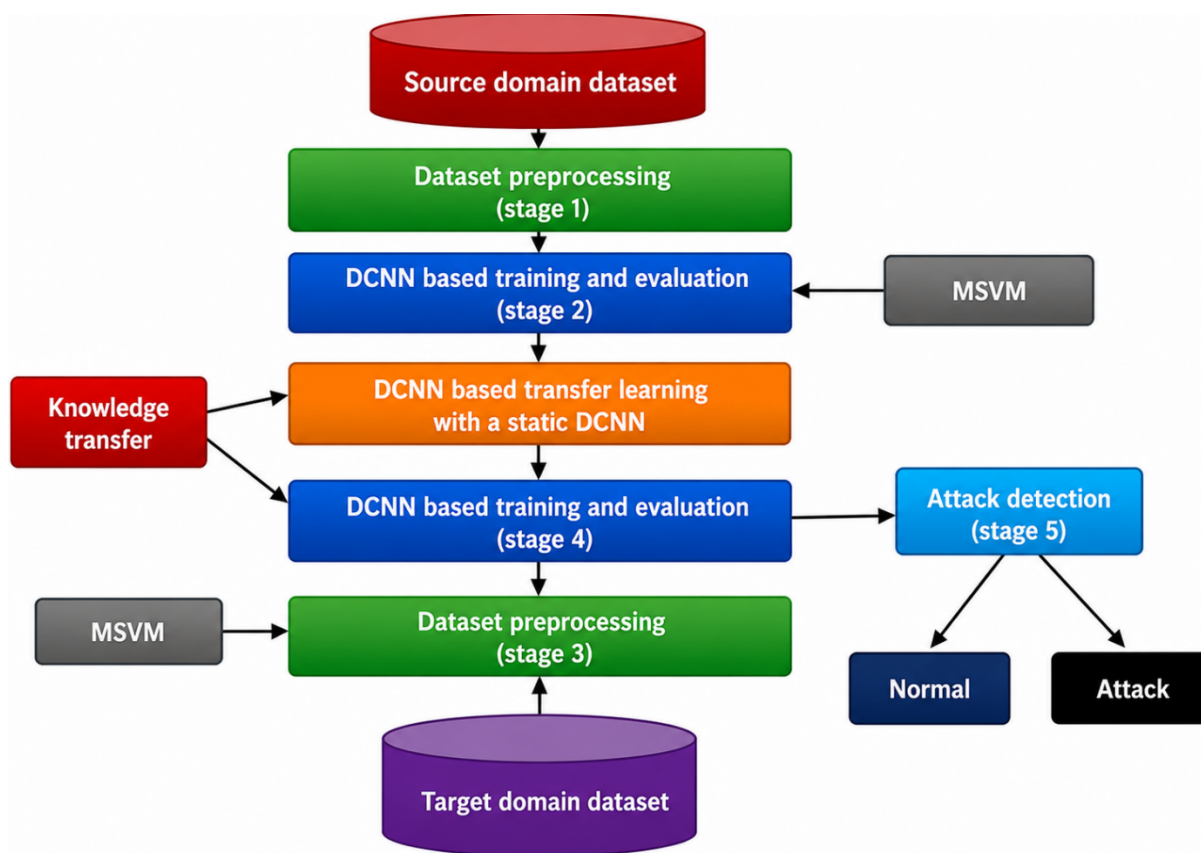


Figure 2. Overall architecture of the proposed intrusion detection framework
Abbreviations: DCNN: Deep convolutional neural network; MSVM: Multi-class support vector machine.

around zero, since it is common practice when dealing with data of different magnitudes. Finally, the standardization function performed standard normalization on selected fields, whereby features were scaled such that they have a mean of zero and a variance of one. These preprocessing steps collectively enhanced the efficiency and effectiveness of intrusion detection models, thus contributing toward the overall goal of improving cyberattack detection in IoT environments.

4.2. Transfer learning

For the IDS framework, the transfer learning-based architecture comprised Stages 2, 4, and 5, with two main phases: first, the creation of an IDS model in the source domain; and second, the adaptation of the model to the target domain through knowledge transfer from the base model.

4.3. Training phase: Source domain

The DCNN-B model has convolutional, pooling, and

flattening layers, and a fully connected layer with softmax activation for classification. This hybrid multi-class SVM (MSVM)-based approach was used by the model to detect multi-class attacks during both the training and testing phases. The accuracy of this model was evaluated on a validation set from the source domain (BoT-IoT). Figure 3 illustrates the architecture of DCNN-B.

The specific details of the DCNN-B model algorithm are shown in Algorithm 2.

A foundational intrusion detection model was established within the source domain to serve as the first line of defense against unauthorized access attempts, malicious activities, and potential security breaches. The main aim was to construct an effective and reliable IDS capable of monitoring and analyzing abnormal patterns in network traffic within the source domain.

The provided set of functions and algorithms outlines a regularization strategy that uses L1 (Lasso) and L2 (Ridge) regularization, implemented in DCNN architectures for

Algorithm 1. Preprocessing of Stages 1 and 3

```
# Input: List of records where each record is a dictionary representing a data entry.
# nominal_fields: List of field names that are nominal and need one-hot encoding.
# Output: encoded_data: List of records with one-hot encoding applied to the nominal fields.
def one_hot_encoding(input_data, nominal_fields):
    encoded_data = []
    for record in input_data:
        encoded_record = create_empty_record()
        for field in nominal_fields:
            field_value = record[field]
            unique_values = get_unique_values(field)
            index = find_index(unique_values, field_value)
            set_bit(encoded_record[field], index)
        append_non_nominal_fields(encoded_record, record, nominal_fields)
        append_to_encoded_data(encoded_data, encoded_record)
    return encoded_data

# Input: input_data: List of records where each record is a dictionary representing a data entry.
# hexadecimal_fields: List of field names that are in hexadecimal format and need to be converted to decimal.
# Output: processed_data: List of records with hexadecimal fields converted to decimal.
def hexadecimal_to_decimal(input_data, hexadecimal_fields):
    processed_data = []
    for record in input_data:
        for field in hexadecimal_fields:
            record[field] = convert_hex_to_decimal(record[field])
        append_to_processed_data(processed_data, record)
    return processed_data

# Input: input_data: List of records where each record is a dictionary representing a data entry.
# logarithmic_fields: List of field names that need logarithmic transformation.
# Output: processed_data: List of records with logarithmic transformation applied to specified fields.
def logarithmic_transformation(input_data, logarithmic_fields):
    processed_data = []
    for record in input_data:
        for field in logarithmic_fields:
            record[field] = apply_logarithmic_transformation(record[field])
        append_to_processed_data(processed_data, record)
    return processed_data

# Input: input_data: List of records where each record is a dictionary representing a data entry.
# standardization_fields: List of field names that need standardization.
# Output: processed_data: List of records with standardization applied to specified fields.
def standardization(input_data, standardization_fields):
    processed_data = []
    for record in input_data:
        for field in standardization_fields:
            record[field] = standardize_field(record[field])
        append_to_processed_data(processed_data, record)
    return processed_data
```

intrusion detection. The `l1_regularization_loss` function was used to compute L1 regularization loss, while the `l2_regularization_loss` function was used to compute L2 regularization loss. These computations depend on the model coefficients as well as the regularization parameters. The comprehensive `total_loss` function combines these two types of losses with the original loss, forming a single regularization term.

In terms of structure, both the DNN and DCNN algorithms share a common framework that includes

weight initialization, propagation through layers with activation functions, max pooling, flattening, and fully connected layers. Importantly, these two methods also incorporate L1 and L2 regularization terms as part of their design. The aim of such an approach is to prevent overfitting while improving model generalization.

The training method operated over epochs, continuously adjusting weights based on the calculated loss and systematically evaluating model performance on validation and test sets. This is important because it aligns

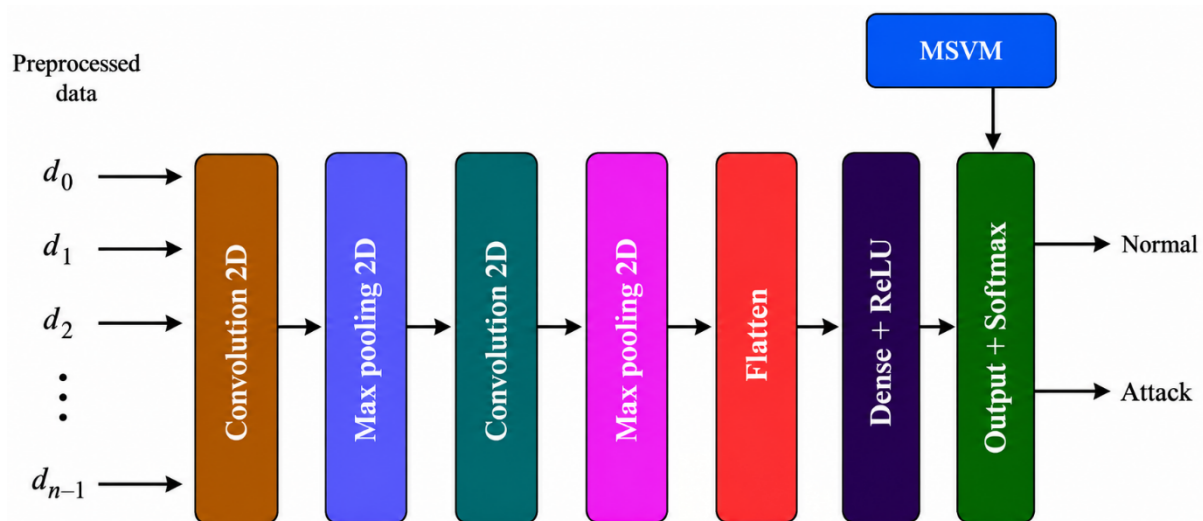


Figure 3. Deep convolutional neural network-B-based intrusion detection system model structure
Abbreviation: MSVM: Multi-class support vector machine.

with the overall goal of improving intrusion detection accuracy by including L1 and L2 regularization terms in the loss function.

4.4. Transfer learning phase

The DCNN-TL model used a frozen convolutional base to retain knowledge from the source domain. During model retraining, this fixed convolutional base ensures that the weights remain unchanged, preventing the loss of valuable learned features. The architecture of the DCNN-TL model, shown in Figure 4, has an unmodified, inherited convolutional base from the DCNN-B model, followed by a sequence of concatenated fully connected layers that form the output layer. The main objective was to adapt the model to the target domain (UNSW-NB15-Basic-Train dataset) without modifying the convolutional base.

The specific details of the DCNN-TL model algorithm are shown in Algorithm 3.

Algorithm 3 is a step-by-step guide for building a transfer learning-based architecture called DCNN-TL, which utilizes a pretrained convolutional base from the DCNN-B model. The process begins by initializing a new sequential model and freezing the pretrained convolutional base of the DCNN-B model, except for the final layer, to preserve knowledge learned from the source domain while enabling adaptation to the target domain. The frozen convolutional base serves as a feature extractor, followed by additional fully connected (dense) layers to facilitate task-specific learning. The architecture of these dense

layers, including the number of units and the use of the ReLU activation function, can be customized according to the target application. Dropout layers are inserted between dense layers to reduce the risk of overfitting.

For each class in the problem statement (e.g., intrusion types), the output layer has a number of units corresponding to the classes, and the softmax activation is used to generate a probability distribution. The model is then compiled by specifying key parameters such as optimizer (eg: Adam), loss function (e.g., sparse categorical cross-entropy), and evaluation metric (e.g., accuracy). The resulting compiled DCNN-TL model can be trained and deployed for intrusion detection tasks within the target domain.

4.5. Multi-class support vector machine

In the proposed framework, the DCNN was used to learn a hierarchy of representations for network traffic using an automatic feature extractor. An MSVM was added as the final classifier, instead of using the traditional softmax layer for classification. The underlying idea of this hybrid architecture is that the deep features generated by the DCNN are highly discriminative, and the MSVM draws optimal decision boundaries which work well for separating multiple attack classes. This combination increases the classifier's ability to differentiate among regular traffic, known attacks, and previously unknown zero-day attacks. Additionally, MSVM performs better on small or even moderately imbalanced datasets, which is relevant to IoT IDS, where high-quality attack samples

are hard to come by. Therefore, a DNN-based multi-class intrusion detection approach is proposed by integrating the feature-extraction capabilities of DNNs with the powerful classification capabilities of SVMs.

Algorithm 2. The deep convolution neural network-B-based multi-class support vector machine model

```
L1 Regularization (Lasso)
function l1_regularization_loss(coefficients, lambda):
    l1_loss = lambda * sum(abs(coefficients))
    return l1_loss
L2 Regularization (Ridge)
function l2_regularization_loss(coefficients, lambda):
    l2_loss = lambda * sum(coefficients^2)
    return l2_loss
function total_loss(data, labels, model_coefficients, lambda1,
lambda2):
    original_loss = compute_original_loss(data, labels, model_
coefficients)
    l1_loss = l1_regularization_loss(model_coefficients, lambda1)
    l2_loss = l2_regularization_loss(model_coefficients, lambda2)
    total_loss = original_loss + l1_loss + l2_loss
    return total_loss
# Algorithm 1: DCNN Algorithm
# Input: I = [X1, X2, ..., Xn]
# Output: Classify Attacks W
# Initialize weight parameters W for each layer l
for each layer l:
    initialize W[l]
# Training loop
for each epoch e in [1, E]:
    # Shuffle the training set
    shuffle training_set
    # Iterate over each training example (xi, yi)
    for each training example (xi, yi):
        # Forward propagation
        # Convolution layer
        z[l] = W[l] * a[l-1] + b[l]
        # Activation layer using ReLU
        a[l] = ReLU(z[l])
        # Max-pooling layer
        a[l] = maxpool(a[l-1])
    a[L] = flatten(a[L])
    # Fully connected layer with ReLU
    z[L+1] = W[L+1] * a[L] + b[L+1]
    # L1 Regularization (Equation 1)
    L1_loss = lambda1 * sum(abs(W))
    # L2 Regularization (Equation 2)
    L2_loss = lambda2 * sum(W^2)
    # Softmax activation
    a[L+1] = softmax(z[L+1])
    # Calculate Loss (Equation 3)
    loss = compute_loss(a[L+1], yi) + L1_loss + L2_loss
    # Update weight parameters
    update_weights(loss)
    # Test model performance on the validation set
```

(Cont'd...)

Algorithm 2. (Continued)

```
test_model(validation_set)
# Test model performance on the test set
test_model(test_set)
# DNN Algorithm
# Input: I = [X1, X2, ..., Xn]
# Output: Classify attacks W
# Initialize weights and biases W^(l), b^(l) for l in 1, 2, ..., L randomly
for each layer l:
    initialize W^(l), b^(l)
# Training loop
for epoch in range(epochs):
    # Iterate over each layer l
    for each layer l:
        # Fully connected layer with ReLU
        z^(l+1) = W^(l+1) * a^(l) + b^(l+1)
        a^(l+1) = ReLU(z^(l+1))
        # L1 Regularization (Equation 1)
        L1_loss = lambda1 * sum(abs(W))
        # L2 Regularization (Equation 2)
        L2_loss = lambda2 * sum(W^2)
        # Calculate Loss (Equation 4)
        loss = compute_loss(a^(L+1), y) + L1_loss + L2_loss
        # Update weight parameters
        update_weights(loss)
    # Test model performance on the validation set
    test_model(validation_set)
# Test model performance on the test set
test_model(test_set)
**Output layer:**
- Add the output layer:
- Units: Number of classes (for intrusion detection)
- Activation: Softmax
**Compile model:**
- Compile the model:
- Choose optimizer (e.g., Adam)
```

The following steps detail the construction of an MSVM with a non-linear exponential kernel function specifically designed for IoT-based intrusion detection:

- (i) Data preparation: The data for training was prepared by calculating the minimum/maximum values of each attribute in the data matrix.
- (ii) Prior probability computation: The prior probabilities were calculated based on cluster sizes and assigned to help with future calculations.
- (iii) Log-likelihood estimation: The maximum log-likelihood estimation for each training data instance was computed using a formula involving exponential joint density estimation and prior probabilities.
- (iv) Cluster optimization: The estimated clusters were optimized by updating weights and prior probabilities according to log-likelihood estimations.
- (v) Model training: Finally, the MSVM model was trained

Algorithm 3. The deep convolution neural network-TL-based multi-class support vector machine model

```

# Input: List of records where each record is a dictionary representing a data entry.
# nominal_fields: List of field names that are nominal and need one-hot encoding.
# Output: encoded_data: List of records with one-hot encoding applied to the nominal fields.
def one_hot_encoding(input_data, nominal_fields):
    encoded_data = []
    for record in input_data:
        encoded_record = create_empty_record()
        for field in nominal_fields:
            field_value = record[field]
            unique_values = get_unique_values(field)
            index = find_index(unique_values, field_value)
            set_bit(encoded_record[field], index)
        append_non_nominal_fields(encoded_record, record, nominal_fields)
        append_to_encoded_data(encoded_data, encoded_record)
    return encoded_data

# Input: input_data: List of records where each record is a dictionary representing a data entry.
# hexadecimal_fields: List of field names that are in hexadecimal format and need to be converted to decimal.
# Output: processed_data: List of records with hexadecimal fields converted to decimal.
def hexadecimal_to_decimal(input_data, hexadecimal_fields):
    processed_data = []
    for record in input_data:
        for field in hexadecimal_fields:
            record[field] = convert_hex_to_decimal(record[field])
        append_to_processed_data(processed_data, record)
    return processed_data

# Input: input_data: List of records where each record is a dictionary representing a data entry.
# logarithmic_fields: List of field names that need logarithmic transformation.
# Output: processed_data: List of records with logarithmic transformation applied to specified fields.
def logarithmic_transformation(input_data, logarithmic_fields):
    processed_data = []
    for record in input_data:
        for field in logarithmic_fields:
            record[field] = apply_logarithmic_transformation(record[field])
        append_to_processed_data(processed_data, record)
    return processed_data

# Input: input_data: List of records where each record is a dictionary representing a data entry.
# standardization_fields: List of field names that need standardization.
# Output: processed_data: List of records with standardization applied to specified fields.
def standardization(input_data, standardization_fields):
    processed_data = []
    for record in input_data:
        for field in standardization_fields:
            record[field] = standardize_field(record[field])
        append_to_processed_data(processed_data, record)
    return processed_data

```

using the MSVM function, utilizing the input data matrix X and the target variable Y . The non-linear exponential kernel function is specified for model training.

The specific details of the MSVM model algorithm are shown in [Algorithm 4](#).

The MSVM model is an approach that helps build an SVM model specifically designed for intrusion detection in a BoT-IoT network. Initially, the process begins with

data preparation, in which attribute statistics, such as minimum and maximum values, are calculated for all features of the dataset. Subsequently, prior probabilities for clusters are computed based on cluster sizes, and log-likelihood estimation is performed for each training data instance using an exponential joint density. Further steps include updating weights and prior probabilities to optimize the estimated clusters, then training an MSVM model with a non-linear exponential kernel. The resulting model can effectively detect and classify intrusions across

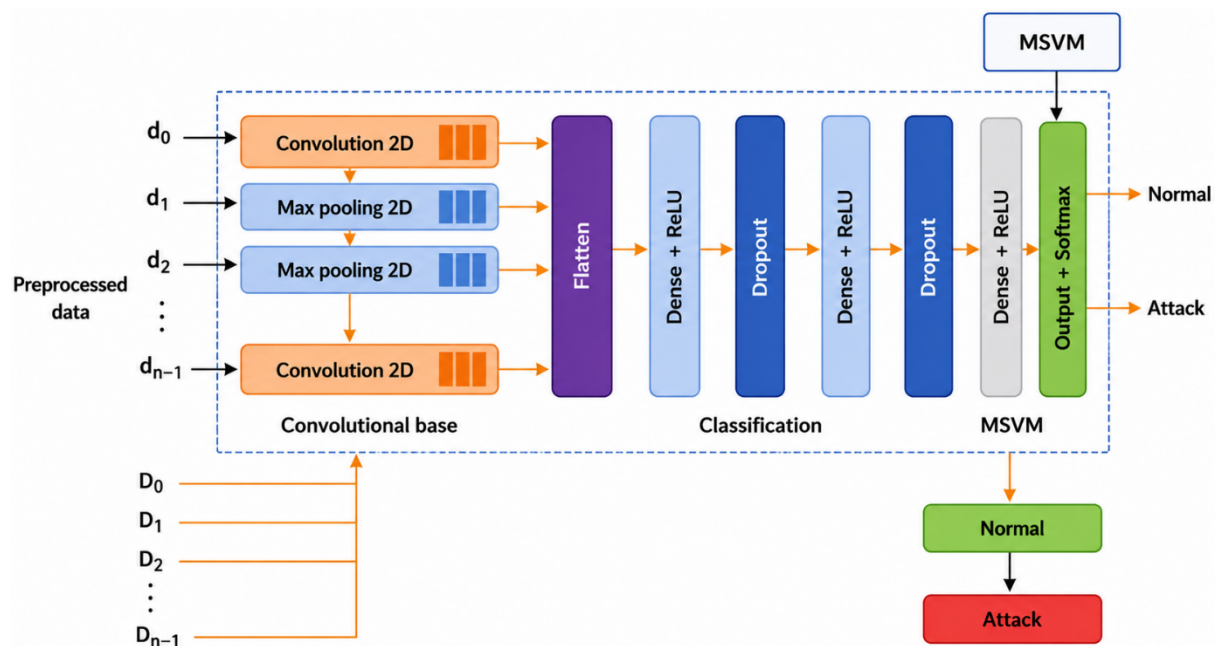


Figure 4. Deep convolution neural network-TL-based intrusion detection system model structure
Abbreviation: MSVM: Multi-class support vector machine.

various types of traffic in an IoT environment by following these steps.

Employing more powerful techniques, such as non-linear exponential kernel functions and log-likelihood estimation, enhances the SVM model, making it strong enough to detect and classify intrusions in IoT botnet networks. The model design ensures that any deviation from normal network behavior can be detected accurately, reinforcing security against cyber threats. This approach emphasizes proactive intrusion detection capabilities in IoT environments, which contribute to resilience against new forms of attack on network systems' integrity. Both MSVM and the final softmax output can be used for classification tasks, and they share similarities and differences in how they operate.

A support vector machine is a supervised learning algorithm used for classification tasks. In multi-class classification problems, its objective is to find an optimal hyperplane that separates data points belonging to one class from those belonging to other classes. An SVM maps input data into a higher-dimensional feature space, where it identifies the optimal hyperplane that maximizes the margin between classes while minimizing classification errors. During prediction, new observations are assigned to a class based on which side of the decision boundary they fall.

Softmax is a function applied to the final layer of neural networks, used mostly in classification models, where raw scores need to be converted into probabilities indicating the likelihood of true events occurring among all possible outcomes considered during training. It ensures that summing these results across many trials yields unity due to the normalization property of softmax outputs, whose denominators are sums, whereas the numerators represent individual occurrences. The output layer assigns a probability to each class using the softmax activation function. The class with the highest probability is selected as the final prediction. Softmax converts the output logits into a probability distribution by applying the exponential function to each activation and then normalizing the results by the sum of all exponentials, yielding values between 0 and 1 that sum to 1.

When comparing SVM and softmax classifiers, distinct characteristics emerge in their decision-making processes. SVMs carve out a discernible decision boundary in the input space, clearly delineating classes, whereas softmax models assign probabilities to each class, conveying nuanced insights into prediction confidence. The interpretability of SVMs stems from their transparent decision boundaries, which facilitate a clear understanding of classification outcomes. In contrast, softmax models provide a probabilistic framework, offering a deeper understanding of the uncertainty associated with

predictions. While SVMs exhibit robustness in handling interpretable decision boundaries, their scalability may falter with large datasets due to the computational burden of quadratic optimization. On the other hand, softmax classifiers, particularly within neural network architectures, demonstrate superior scalability and efficiency, enabling seamless processing of extensive datasets. Moreover, softmax classifiers excel at handling class imbalances through techniques such as class weighting, ensuring equitable treatment of disparate class distributions. Thus, the choice between SVMs and softmax classifiers hinges on factors such as interpretability, scalability, and the nature of the dataset, each offering distinct advantages tailored to specific application domains.

Algorithm 4. The multi-class support vector machine model

1. The multi-class support vector machine (SVM) with a non-linear exponential kernel function algorithm can be mathematically expressed as follows:
2. Data preparation: Let X be the data matrix with M instances and N features; the min and max values of each attribute are computed:
 $X_{\min} = \min(X)$
 $X_{\max} = \max(X)$
3. Prior probability computation: The prior probability of each cluster is computed using the cluster size and assigned to each cluster:
 $\text{InitPriorProb}(k) = C_s / \sum(C_s)$
4. Log-likelihood estimation: The maximum log-likelihood estimation for each training data instance is calculated using the following formula:
 $\exp_{\text{joint_density}} = e^{(-0.5 * (X - CK)^2 / \sigma^2)}$
 $\log_likelihood = \log(\exp_{\text{joint_density}} * \text{InitPriorProb}(k))$
5. Cluster optimization: The estimated clusters are optimized by updating the weights and prior probabilities using the following formula:
 $\text{updated_weights} = \log_likelihood / \sum(\log_likelihood)$
 $\text{updated_prior_prob} = \text{updated_weights} / \sum(\text{updated_weights})$
6. Model training: The final step involves training the MSVM model with a non-linear exponential kernel function:
 $\text{model} = \text{MSVM}(X, Y, \text{kernel} = \text{"nonlinexp"})$
 where Y is the target variable indicating the class of each instance, and $\text{kernel} = \text{"exp"}$ specifies the use of a non-linear exponential kernel function.

Both methods can be compared using metrics such as accuracy, precision, recall, and F1 score on a validation or test set. The class predicted by the model (SVM) with the highest softmax probability can be used to evaluate performance consistency. At the start and finish of a framework, data preprocessing plays an important role. It uses the convolutional base's output to effectively feed into the classifier, as its model architecture relies on transfer

learning. Thus, both source and target domains require the same input shape and features between datasets to be consistent with each other. To achieve this, revised versions of both datasets were generated, containing only the 15 common features identified. These common features, detailed in Table 2, facilitated uniformity in the input data, enabling seamless training and evaluation of the TL model across domains. By aligning the datasets based on these common features, the framework sets a solid foundation for effective knowledge transfer and adaptation between source and target domains, ultimately enhancing the model's performance and generalization capabilities.

5. Experimentation and results

5.1. Metrics

To evaluate cyberattack detection using transfer learning, we used accuracy, precision, recall, FPR, and F1-score. These were calculated from a confusion matrix, which shows how many instances were classified as which category by the classifier. True positives (TP) and true negatives (TN) refer to correctly identified attack records and normal records, respectively; while false positives (FP) and false negatives (FN) represent normal records misclassified as attacks or vice versa. Accuracy (ACC) is the number of correct predictions divided by the total instances (Equation 1).

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision (P) is the proportion of items predicted correctly to all items predicted (Equation 2).

$$P = \frac{TP}{TP + FP} \quad (2)$$

Recall (R) is the ratio between correctly classified examples and all actual positive examples (Equation 3).

$$r = \frac{TP}{TP + FN} \quad (3)$$

The FPR was calculated as the ratio of items incorrectly classified as positive (attack) to all actual negative instances (normal) (Equation 4).

$$FPR = \frac{FP}{TN + FP} \quad (4)$$

The F1-score balances recall and precision by taking their harmonic mean (Equation 5).

$$FPR = 2 * \frac{(p * r)}{p + r} \quad (5)$$

Table 2. Common features for the BoT-IoT and UNSW-NB15 datasets

BoT-IoT	UNSW-NB15	Type	Description
proto	proto	Nominal	Textual representation of transaction protocols present in network flow
saddr	srcip	Nominal	
sport	sport	Integer	Source port number
daddr	dstip	Nominal	Destination IP address
dport	dsport	Integer	Destination port number
spkts	spkts	Float	Source-to-destination packet count
dpkts	dpkts	Float	Destination-to-source packet count
sbytes	sbytes	Float	Source-to-destination byte count
dbytes	dbytes	Float	Destination-to-source byte count
state	state	Nominal	Transaction state
stime	stime	Timestamp	Record start time
ltime	ltime	Timestamp	Record last time
dur	dur	Float	Record total duration
attack	label	Binary	Class label; 0 for normal traffic, 1 for attack
category	attack_cat	Nominal	Cyberattack family

Abbreviation: UNSW-NB15: University of New South Wales Network-Based 2015 dataset.

5.2. System setup

We used a Lenovo IdeaPad 320-15IKB laptop with an Intel® Core™ i7-8850U CPU @ 1.8 GHz processor and 8 GB RAM for our experiments. To implement a transfer learning-based solution, the TensorFlow backend (Penaganti, 2025) was used with the Keras frontend, and the Pandas and Scikit-learn packages were also employed. The code can be found in the GitHub repository (Bongu, 2025). In this paper, we used the BoT-IoT and UNSW-NB15 datasets to evaluate the effectiveness of the proposed framework.

5.3. Source domain dataset

The Australian Center for Cyber Security developed the BoT-IoT dataset as part of their research work on cybersecurity threats in IoT environments (Satyanarayana & Chatrpathi, 2025). The data represents network activity in a simulated IoT environment comprising both legitimate (benign) and malicious (attack) traffic. The dataset was generated at the Cyber Range Lab of UNSW Canberra and comprises records from 62 hosts on a network identified by the address space 192.168.100.0/26, with each host having 46 features, resulting in five output classes altogether (Gunupusala & Kaila, 2024).

These included normal, DoS, distributed DoS, reconnaissance, and information theft attacks. DoS/distributed DoS types seek to disrupt server operations by flooding them with traffic. Transmission Control

Protocol/User Datagram Protocol/Hypertext Transfer Protocol are some examples of this type, which aim to consume resources such as bandwidth or processing power required for handling legitimate requests; such attacks may be directed at any layer in the Open Systems Interconnection model but mostly happen at layers three through seven (refer to Requests for Comments). For example, an Internet Control Message Protocol flood can be used to render routers incapable of forwarding packets between different networks by targeting their central processing unit utilization, while a synchronize flood attack floods the target system with half-open Transmission Control Protocol connection requests so that it becomes overwhelmed trying to complete them all. Additionally, Hypertext Transfer Protocol GET flood attempts saturate web servers' ability to process client requests for pages by sending multiple requests each requiring a full response; if successful, these could lead to unavailability of service (DoS).

Reconnaissance attacks involve scanning other people's systems/networks for vulnerabilities which can later be exploited during an intrusion detection attempt such as operating system fingerprinting or service scanning. Information theft attacks capture sensitive information like keystrokes typed on a keyboard or data being sent/received over wiretap connections between devices on the same local area network.

The BoT-IoT dataset contains more than 73 million records. To ensure computational efficiency while maintaining data representativeness, only 10% of the dataset was used in this study. The selected subset preserved the distribution of normal traffic and the various attack categories present in the full dataset. Table 3 provides additional details for reference purposes (Satyanarayana & Chatrapathi, 2025).

Table 3. BoT-IoT dataset

Dataset	Normal	Attack	% Attack	% Novel attack
Bot-IoT	9,543	5,823,226	99.84%	-
UNSW-NB15-Basic-Train	217,552	217,967	50.04%	-
UNSW-NB15-Basic-Test	72,794	72,379	49.85%	0.00%
UNSW-NB15-Test+	30,937	30,937	50.00%	100.00%
UNSW-NB15-Test	321,283	321,283	50.00%	9.63%

Abbreviation: UNSW-NB15: University of New South Wales Network-Based 2015 dataset.

5.4. Target domain dataset

Four unique datasets were taken from the UNSW-NB15, which was created by the Australian Center for Cyber Security and researchers from around the globe. It imitates an artificial IoT environment by gathering normal network traffic and synthetically creating attack activity. This dataset includes different types of traffic, such as generic, exploits, fuzzers, DoS, reconnaissance, analysis, backdoors, shellcode, and worms, along with normal traffic that cover a wide range of attack scenarios. The dataset consists of 49 features and is large, with more than 2 million records in four CSV files. However, the distribution between normal traffic and malicious traffic is imbalanced; the former accounts for 87% of the dataset while the latter only makes up 13%. Table 4 provides a detailed description and distribution of the dataset to help readers understand it better (Satyanarayana, 2023).

To test how well the transfer learning-based intrusion detection framework can detect new attacks, three datasets were built within its target domain:

- (i) UNSW-NB15-Basic: This dataset was used as a base to train an initial model for detecting known attacks. It contains normal traffic as well as four specific types

of attack (generic, exploits, DoS, and reconnaissance) chosen from a larger dataset called UNSW-NB15. The dataset was designed to reflect real-world scenarios, so there are equal numbers of normal traffic instances and attack instances within it; 75% was allocated for training (UNSW-NB15-Basic-Train), and 25% was reserved for testing (UNSW-NB15-Basic-Test).

- (ii) UNSW-NB15-Test+: This dataset was used to evaluate the model's ability to detect previously unseen attack families by introducing five novel attack types (fuzzers, analysis, backdoors, shellcode, and worms) into normal traffic that were absent in previous datasets such as UNSW-NB15-Basic.
- (iii) UNSW-NB15-Test: The objective was twofold: to evaluate the framework's ability to detect both known and previously unseen attack categories based on knowledge acquired during training on the nine attack classes included in the UNSW-NB15 dataset.

During the construction of the UNSW-NB15-Basic dataset, the number of normal traffic and attack samples was balanced to create an equal class distribution, providing a well-balanced training set for model development. However, the UNSW-NB15-Test+ and UNSW-NB15-Test sets introduced new attack categories not covered by previous ones, making it more challenging for a transfer learning-based IDS to adapt well to scenarios where novel attack patterns are likely encountered frequently. Further details are provided in Tables 5 and 6, which summarize the composition of the datasets and facilitate interpretation of the evaluation results.

In the source domain, a portion consisting of 10% of the BoT-IoT dataset was used for training and testing purposes; this was randomly split such that 75% went toward training while the remaining portion was reserved solely for testing purposes. The composition between benign/malicious traffic within each part is provided in Table 7. Notably, the attack distribution differed between UNSW-NB15-Test+ and UNSW-NB15-Test; the former focused only on zero-days, whereas the latter incorporated both known and unknown zero-days, allowing us to evaluate our model against different test scenarios where new/known threats may arise.

Two main test scenarios were considered during the performance evaluation phase: (i) zero-day attack detection using the BoT-IoT dataset; and (ii) detecting zero-day attacks using UNSW-NB15.

5.5. Zero-day attack detection using the BoT-IoT dataset

The proposed framework was pretrained on the BoT-IoT dataset, which acts as a source domain prior to any other

Table 4. University of New South Wales Network-Based 2015 dataset

Category	Records	Description
Normal	2,218,761	Natural transaction data
Generic	215,481	Attack against blockciphers with a given block and key size (not considering its structure)
Exploits	44,525	Attack that exploits vulnerabilities, taking advantage of security problems (of an operating system or a piece of software) known by the attackers
Fuzzers	24,246	An attack that suspends a program or network, feeding it with randomly generated data
DoS	16,353	A malicious attack that makes a server or network resource unavailable, overloading the target of the associated infrastructure with a flood of internet traffic
Reconnaissance	13,987	Comprises different attacks that gather information
Analysis	2677	Different attacks on penetrations (HTML files, spam, and port scan)
Backdoors	2329	An attack that bypasses a system security mechanism to access a computer or its data
Shellcode	1511	Attack that exploits software vulnerabilities using small pieces of code as payloads
Worms	174	Attack where the attacker replicates itself to spread to other computers

Table 5. UNSW-NB15-Basic-Train and UNSW-NB15-Basic-Test datasets

Name	UNSW-NB15-Basic-Train		UNSW-NB15-Basic-Test	
	Records	Percentage	Records	Percentage
Normal	217,552	49.95%	72,794	50.14%
Generic	161,865	37.17%	53,616	36.93%
Exploits	33,408	7.67%	11,117	7.66%
Denial of service	12,196	2.80%	4,157	2.86%
Reconnaissance	10,498	2.41%	3,489	2.40%

Abbreviation: UNSW-NB15: University of New South Wales Network-Based 2015 dataset.

Table 6. UNSW-NB15-Test+ and UNSW-NB15-Test datasets

Name	UNSW-NB15-Test+		UNSW-NB15-Test	
	Records	Percentage	Records	Percentage
Normal	30,937	50.00%	321,283	50.00%
Generic	-	-	215,481	33.53%
Exploits	-	-	44,525	6.93%
Denial of service	-	-	16,353	2.54%
Reconnaissance	-	-	13,987	2.18%
Fuzzers	24,246	39.19%	24,246	3.77%
Analysis	2,677	4.33%	2,677	0.42%
Backdoor	2,329	3.76%	2,329	0.36%
Shellcode	1,511	2.44%	1,511	0.24%
Worms	1,74	0.28%	1,74	0.03%

Abbreviation: UNSW-NB15: University of New South Wales Network-Based 2015 dataset.

form of training taking place. The DCNN with MSVM (DCNN-B) was trained over 35 epochs using a batch size of 2,048. An Adam optimizer was employed with a learning rate of 0.94528 to minimize the error function, and categorical cross-entropy loss was used for guidance.

Subsequently, the UNSW-NB15-Basic-Train dataset was used in transfer learning, representing the target domain. The DCNN-TL model was trained for 35 epochs with a batch size of 5,126 during transfer learning. The Adam optimizer was chosen for effective transfer learning with a lower learning rate of 0.94528; however, the categorical cross-entropy loss function was still employed. A lower learning rate during transfer learning enables more gradual fine-tuning of the pretrained model, improving its adaptation to the target domain and enhancing overall performance. The training parameters used in the proposed framework are summarized in Table 8.

5.6. Zero-day attack detection using the University of New South Wales Network-Based 2015 dataset

Two different datasets were considered when testing

the transfer learning solution: the UNSW-NB15-Test+ dataset, which contains only zero-day attacks, and the UNSW-NB15-Test dataset, where known attacks are mixed with these types. Initially, the efficiency of our transfer learning model in detecting zero-days was assessed using the UNSW-NB15-Test+ dataset. It showed remarkable performance metrics such as 99.51% accuracy, 99.56% precision, 99.51% recall, 0.03% FPR, and 99.45% F1-score. These results indicate how robust this model can be at identifying zero-day attacks accurately. Table 9 also describes the number of samples detected (TPs), not detected (FNs), and false alarms (FPs), as well as detection rates for each of the five different zero-day attack examples according to their features. Notably, all these attack categories have a detection rate greater than 98%, which implies high effectiveness against various types of such threats by our proposed approach.

Moreover, this same transfer learning model was evaluated for detection of both known and zero-day attacks using the UNSW-NB15-Test dataset. In this comprehensive evaluation, the DCNN-TL model achieved

Table 7. Dataset summary showing the number of records corresponding to normal and malicious traffic

Dataset	Normal	Attack	% Attack	% Novel attack
Bot-IoT	9,543	5,823,226	99.84%	-
UNSW-NB15-Basic-Train	217,552	217,967	50.04%	-
UNSW-NB15-Basic-Test	72,794	72,379	49.85%	0.00%
UNSW-NB15-Test+	30,937	30,937	50.00%	100.00%
UNSW-NB15-Test	321,283	321,283	50.00%	9.63%

Abbreviation: UNSW-NB15: University of New South Wales Network-Based 2015 dataset.

Table 8. Transfer learning model training parameters summary

Model	Epochs	Batch size	Optimizer	Learning rate α	Loss
DCNN-B	35	2,048	Adam	0.94528	Categorical cross-entropy
DCNN-TL	25	4,096	Adam	0.94528	Categorical cross-entropy

Abbreviations: DCNN: Deep convolutional neural network; TL: Transfer learning.

Table 9. Attack detection summary of the University of New South Wales Network-Based 2015 dataset: Zero-day attacks

Traffic	Detection rate	Detected samples	Non-detected samples
Normal	98.14%	30,350	521
Analysis	100.00%	622	0
Backdoor	100.00%	357	0
Fuzzers	99.96%	21,508	9
Shellcode	99.93%	1,510	1
Worms	98.85%	172	2

strong metrics: 99.59% accuracy, 99.52% precision, 99.79% recall, FPR=0.04%, and F1-score=98.97%. Although performance decreased slightly compared with the evaluation focused solely on zero-day attacks, the proposed framework maintained strong detection performance on previously unseen attack types. Table 10 presents the family-wise detection rates for the different zero-day attack categories, demonstrating the framework's effectiveness in real-world intrusion detection scenarios, where novel attacks may emerge unexpectedly.

To enable comparison between our proposed transfer learning-based solution and existing DL-based IDSs, we implemented a DCNN-based solution which was then evaluated against the performance of our transfer learning-based approach.

Firstly, we ran the DCNN model to see if it could detect any zero-days using the NSW-NB15-Test+ dataset, where it recorded an accuracy of 67.84%, a precision value of 80.77%, a recall rate of 79.92%, an FPR of 68.56%, and an F1-score of up to 69.25%. Subsequently, the same dataset was used again and included all types of attacks, such as known ones and those from the previous step. Under these conditions, the proposed framework achieved an accuracy of 87.26%, a precision of 93.35%, and a recall of 87.43%, while the FPR decreased substantially to 12.73%, demonstrating a significant reduction in false alarms compared with the baseline DCNN model.

When all metrics were compared, it is clear that the transfer learning model performed better than the DCNN model. Notably, the transfer learning model improved detection accuracy by 19.42 percentage points for zero-day attacks and by 14.61 percentage points for the combined detection of known and zero-day attacks compared

with the baseline model. These improvements are likely attributable to the use of pretrained weights from a related source domain, which enable the transfer learning model to converge more rapidly toward an optimal solution than the DCNN model initialized with random weights.

Another measure showing significant improvement after adoption of the transfer learning approach is FPR. The reason a transfer learning-based IDS has lower FPR values can be attributed to two factors. First, it is due to class imbalance caused mainly by the occurrence of rare events such as zero-day attacks. Secondly, it is due to the inherent nature of these intrusions, as they remain unknown until detected, making them difficult or impossible for any algorithm to learn during the training phase unless they are specifically programmed. Thus, during later stages, when such attacks occur, the model can identify and classify them accordingly.

Moreover, we analyzed the detection rate per family, comparing CNN-based IDS and transfer learning-based IDS when only zero-day attacks were considered or when both known and zero-day attacks were included. The outcome showed higher detection rates across all types of attacks by the transfer learning-based system, whose values ranged from 99.98% up to 100%. In addition, the detection rate for zero-day attacks improved in the transfer learning-based approach compared with CNN, where it reached 19.73%, hence showing how effective CNNs can be at recognizing them despite having very little information about their existence within the dataset used during the training phase. Moreover, known attacks were not left out either, achieving a maximum increase of about 27.69%, as shown in Table 11. Therefore, transfer learning is indeed more effective than CNN for this purpose (Figures 5 and 6).

Table 10. Attack detection summary of the University of New South Wales Network-Based 2015 dataset: Known and zero-day attacks

Traffic	Detection rate	Detected samples	Non-detected samples
Normal	98.14%	315,902	46,081
Denial of service	99.56%	3,841	22
Exploits	99.76%	28,249	68
Generic	99.98%	213,678	40
Reconnaissance	99.95%	11,848	6
Analysis	99.84%	621	1
Backdoor	99.44%	355	2
Fuzzers	99.89%	21,472	45
Shellcode	99.95%	1,510	1
Worms	98.88%	172	2

Table 11. Attack detection rate for known and zero-day attacks

Traffic	UNSW-NB15-Test			UNSW-NB15-Test+		
	DCNN	TL	Improvement	DCNN	TL	Improvement
Normal	99.65%	98.14%	-1.51%	99.72%	98.35%	-1.37%
DoS	95.63%	99.56%	3.93%	-	-	-
Exploits	96.80%	99.76%	2.96%	-	-	-
Generic	97.13%	99.98%	2.85%	-	-	-
Reconnaissance	93.67%	99.95%	6.28%	-	-	-
Analysis	89.25%	99.84%	10.59%	77.86%	99.78%	21.92%
Backdoor	88.41%	99.44%	11.03%	87.53%	99.90%	12.37%
Fuzzers	80.16%	99.89%	19.73%	72.26%	99.95%	27.69%
Shellcode	90.42%	99.95%	9.53%	98.32%	99.13%	0.81%
Worms	98.41%	99.88%	1.47%	96.97%	98.82%	1.85%

Abbreviations: DCNN: Deep convolutional neural network; TL: Transfer learning; UNSW-NB15: University of New South Wales Network-Based 2015 dataset.

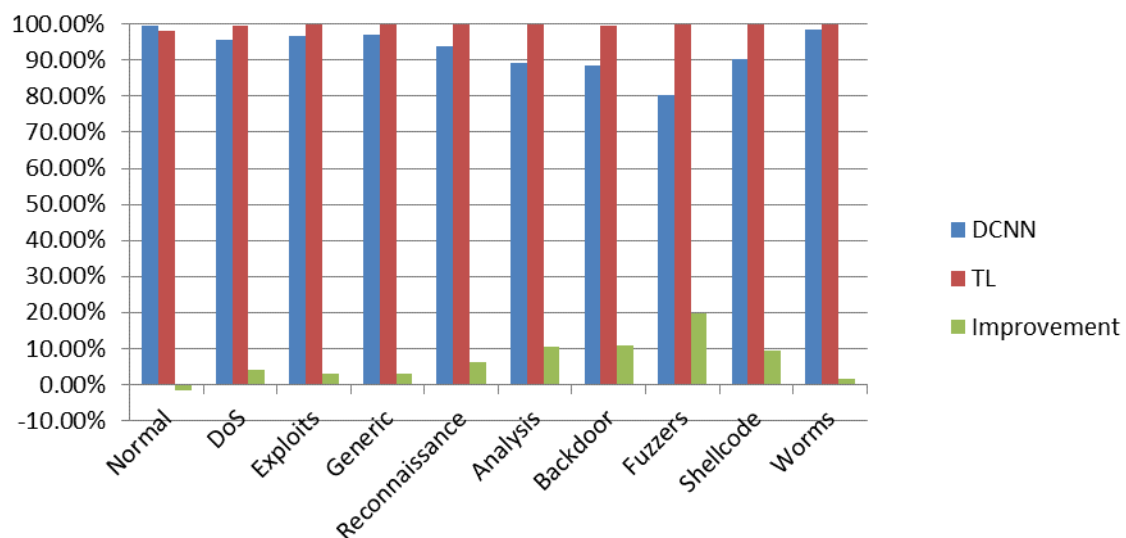


Figure 5. Comparison of the DCNN and the proposed transfer learning model trained on the UNSW-NB15 and evaluated on the UNSW-NB15-Test dataset

Abbreviations: DCNN: Deep convolutional neural network; DoS: Denial of service; TL: Transfer learning; UNSW-NB15: University of New South Wales Network-Based 2015 dataset.

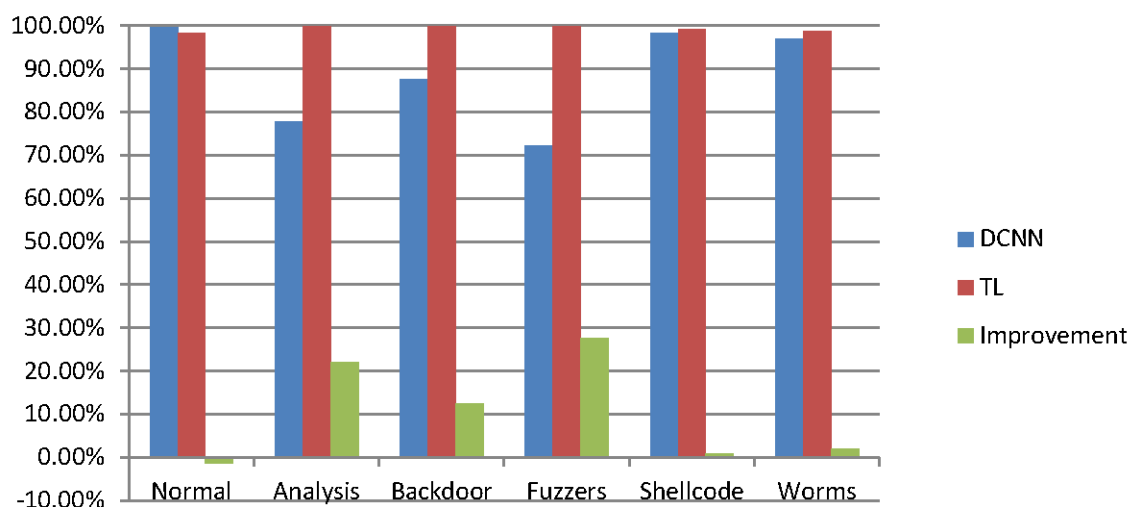


Figure 6. Comparison of the DCNN and the proposed transfer learning model trained on the BoT-IoT dataset and evaluated on the UNSW-NB15-Test+ dataset

Abbreviations: DCNN: Deep convolutional neural network; IoT: Internet of things; TL: Transfer learning; UNSW-NB15: University of New South Wales Network-Based 2015 dataset.

6. Conclusion

We have spent considerable effort investigating how transfer learning-based IDSs can be useful for detecting zero-day attacks in IoT networks with limited and imbalanced datasets. We also developed an IDS capable of detecting both known cyberattack families and emerging attack types with a high level of accuracy. To realize this, we combined knowledge transfer with model refinement into one effective framework for intrusion detection, which showed good results against all types of cyberattacks, whether they are familiar or strange (novel). As a case study, the BoT-IoT dataset was used to obtain knowledge (source domain), which was then applied to the UNSW-NB15 (target domain). To check if these systems could detect unknown attacks, we designed a test set containing five different types of zero-day attacks. Our findings reveal that even when there is a lack of balance in the dataset, but enough labeled data for zero-day attack identification is provided, fine-tuning along with network transfer improves the performance of IDS greatly such that its FPR becomes almost non-existent while still maintaining a very high level of accuracy, especially on these novel events. Experimental results indicate that great accuracy rates can be achieved by a transfer learning-based method with extremely low FPRs; moreover, detection rates also increase significantly over various families, both known and zero-day attacks, compared to earlier methods used by DL-based IDSs where lightweight devices were employed under authentic traffic conditions from IoT networks using the UNSW-NB15.

Acknowledgments

None.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Author contributions

Conceptualization: Pasi Ashok Kumar

Data curation: Mohammad Sirajuddin

Formal analysis: Gunupusala Satyanarayana, Nimmala Mangathayaru

Investigation: Gunupusala Satyanarayana, Mohammad Sirajuddin, Nimmala Mangathayaru

Methodology: Gunupusala Satyanarayana, Pasi Ashok Kumar

Project administration: Emandi Sreedevi, Jhansi Lakshmi Sarwani Theeparthi, Gunupusala Satyanarayana

Resources: Nimmala Mangathayaru, Jhansi Lakshmi Sarwani Theeparthi, Gunupusala Satyanarayana

Software: Gunupusala Satyanarayana, Nimmala Mangathayaru

Supervision: Emandi Sreedevi, Jhansi Lakshmi Sarwani Theeparthi, Pasi Ashok Kumar

Validation: Mohammad Sirajuddin, Nimmala Mangathayaru, Emandi Sreedevi, Pasi Ashok Kumar
Visualization: Mohammad Sirajuddin
Writing–original draft: Gunupusala Satyanarayana
Writing–review & editing: Mohammad Sirajuddin, Emandi Sreedevi, Pasi Ashok Kumar

Availability of data

Data are available from the corresponding author upon request.

References

- Aceto, G., Ciunzo, D., Montieri, A., & Pescapé, A. (2020). Toward effective mobile encrypted traffic classification through deep learning. *Neurocomputing*, 409, 306–315.
<https://doi.org/10.1016/j.neucom.2020.05.036>
- Bilge, L., & Dumitras, T. (2012). Before we knew it: An empirical study of zero-day attacks in the real world. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, October 16–18, 2012, Raleigh, NC. 833–844.
<https://doi.org/10.1145/2382196.2382284>
- Bongu, S. R. (2025). Real-Time Behavioral Biometrics and Continuous Authentication Framework for Secure Financial Transaction Ecosystems. *Journal of Applied Sciences and Modelling*, 1(1), 40–50.
<https://doi.org/10.71426/jasm.v1.i1.pp40-50>
- Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176.
<https://doi.org/10.1109/comst.2015.2494502>
- Chapman, C. (2016). Chapter 1—Introduction to Practical Security and Performance Testing. In Chapman, C. (Ed.) *Network Performance and Security*. Boston, MA: Syngress. (pp. 1–14).
<https://doi.org/10.1016/B978-0-12-803584-9.00001-9>
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.
<https://doi.org/10.1613/jair.953>
- Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., & Mizutani, K. (2017). State-of-the-Art Deep Learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Communications Surveys & Tutorials*, 19(4), 2432–2455.
<https://doi.org/10.1109/comst.2017.2707140>
- Fan, Y., Li, Y., Zhan, M., Cui, H., & Zhang, Y. (2020). IoTDefender: A Federated Transfer Learning Intrusion Detection Framework for 5G IoT. In: *Proceedings of the 2020 IEEE International Conference on Big Data Science and Engineering (BigDataSE)*, December 29, 2020–January 1, 2021, Guangdong, China. 88–95.
<https://doi.org/10.1109/BigDataSE50710.2020.00020>
- Ficke, E., Schweitzer, K., Bateman, R., & Xu, S. (2019). Analyzing Root Causes of Intrusion Detection False-Negatives: Methodology and Case Study. In: *Proceedings of the 2019 IEEE Military Communications Conference (MILCOM)*, November 12–14, 2019, Norfolk, VA. 1–6.
<https://doi.org/10.1109/MILCOM47813.2019.9020860>
- Gopalakrishnan, K., Khaitan, S. K., Choudhary, A., & Agrawal, A. (2017). Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and Building Materials*, 157, 322–330.
<https://doi.org/10.1016/j.conbuildmat.2017.09.110>
- Guan, J., Cai, J., Bai, H., & You, I. (2021). Deep transfer learning-based network traffic classification for scarce dataset in 5G IoT systems. *International Journal of Machine Learning and Cybernetics*, 12(11), 3351–3365.
<https://doi.org/10.1007/s13042-021-01415-4>
- Gunupusala, S., & Kaila, S. C. (2024). Multi-Class network anomaly detection using machine learning techniques. *Contemporary Mathematics*, 5(2), 37–49.
<https://doi.org/10.37256/cm.5220243723>
- Hamed, T., Ernst, J. B., & Kremer, S. C. (2018). A survey and taxonomy of classifiers of intrusion detection systems. In Beyah, B., Chang, R., Li, Y., & Zhu, S. (Eds.) *Computer and Network Security Essentials*. Cham, Switzerland: Springer. (pp. 21–69).
https://doi.org/10.1007/978-3-319-58424-9_2
- Hindy, H., Hodo, E., Bayne, E., Seeam, A., Atkinson, R., & Bellekens, X. (2018). A taxonomy of malicious traffic for intrusion detection systems. *arXiv*. arXiv:1806.03516.
<https://doi.org/10.48550/arXiv.1806.03516>
- Hindy, H., Brosset, D., Bayne, E., Seeam, A. K., Tachtatzis, C., Atkinson, R., & Bellekens, X. (2020). A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access*, 8, 104650–104675.
<https://doi.org/10.1109/ACCESS.2020.3000179>
- Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., & Atkinson, R. (2017). Shallow and deep networks intrusion detection system: A taxonomy and survey. *arXiv*. arXiv:1701.02145.
<https://doi.org/10.48550/arXiv.1701.02145>
- Idrissi, I., Azizi, M., & Moussaoui, O. (2021). Accelerating the update of a DL-based IDS for IoT using deep transfer learning. *Indonesian Journal of Electrical Engineering and Computer Science*, 23(2), 1059–1067.

- <https://doi.org/10.11591/ijeecs.v23.i2.pp1059-1067>
- Kaloudi, N., & Li, J. (2021). The AI-Based cyber threat landscape. *ACM Computing Surveys*, 53(1), 1–34.
- <https://doi.org/10.1145/3372823>
- Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1), 1–22.
- <https://doi.org/10.1186/s42400-019-0038-7>
- Kilincer, I. F., Ertam, F., & Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks*, 188, 107840.
- <https://doi.org/10.1016/j.comnet.2021.107840>
- Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, 100, 779–796.
- <https://doi.org/10.1016/j.future.2019.05.041>
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. *Nature*, 521(7553), 436–444.
- <https://doi.org/10.1038/nature14539>
- Masum, M., & Shahriar, H. (2020, December 8–10). TL-NID: Deep Neural Network with Transfer Learning for Network Intrusion Detection. In: *Proceedings of the 2020 15th International Conference for Internet Technology and Secured Transactions (ICITST)*, Online. 1–7.
- <https://doi.org/10.23919/ICITST51030.2020.9351317>
- Maz, Y. A., Anbar, M., Manickam, S., Abualhaj, M. M., Almalki, S. A., & Alabsi, B. A. (2025). Transfer Learning-Based Approach with an Ensemble Classifier for Detecting Keylogging Attack on the Internet of Things. *Computers, Materials & Continua*, 85(3), 5287–5307.
- <https://doi.org/10.32604/cmc.2025.068257>
- Mehedi, S. T., Anwar, A., Rahman, Z., & Ahmed, K. (2021). Deep Transfer learning based intrusion detection system for electric vehicular networks. *Sensors*, 21(14), 4736.
- <https://doi.org/10.3390/s21144736>
- Mehedi, S. T., Anwar, A., Rahman, Z., Ahmed, K., & Islam, R. (2023). Dependable Intrusion Detection System for IoT: A Deep Transfer Learning Based Approach. *IEEE Transactions on Industrial Informatics*, 19(1), 1006–1017.
- <https://doi.org/10.1109/tii.2022.3164770>
- Metrick, K., Najafi, P., & Semrau, J. (2020). Zero-Day Exploitation Increasingly Demonstrates Access to Money, Rather than Skill—Intelligence for Vulnerability Management (Part One). Technical Report, FireEye Technical Report. Accessed 10 June 2021: <https://www.fireeye.com/blog/threat-research/2020/04/zero-day-exploitation-demonstrates-access-to-money-not-skill>.
- Morid, M. A., Borjali, A., & Del Fiol, G. (2021). A scoping review of transfer learning research on medical image analysis using ImageNet. *Computers in Biology and Medicine*, 128, 104115.
- <https://doi.org/10.1016/j.compbiomed.2020.104115>
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, November 10–12, 2015, Canberra, Australia. 1–6.
- <https://doi.org/10.1109/MilCIS.2015.7348942>
- Nguyen, T. T., & Reddi, V. J. (2023). Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8), 3779–3795.
- <https://doi.org/10.1109/TNNLS.2021.3121870>
- Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345–1359.
- <https://doi.org/10.1109/tkde.2009.191>
- Penaganti, R. (2025). Graph Neural Network-Based Framework for Real-Time Financial Fraud Detection in Digital Payment Ecosystems. *Journal of Computing and Data Technology*, 1(2), 91–97.
- <https://doi.org/10.71426/jcdt.v1.i2.pp91-97>
- Rodriguez, E., Otero, B., Gutierrez, N., & Canal, R. (2021). A survey of deep learning techniques for cybersecurity in mobile networks. *IEEE Communications Surveys & Tutorials*, 23(3), 1920–1955.
- <https://doi.org/10.1109/comst.2021.3086296>
- Ruder, S., Peters, M. E., Swayamdipta, S., & Wolf, T. (2021). Transfer Learning in Natural Language Processing. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials (NAACL-HLT Tutorials)*, June 2–7, 2021, Minneapolis, MN. 15–18.
- <https://doi.org/10.18653/v1/N19-5004>
- Sameera, N., & Shashi, M. (2019). Transfer Learning Based Prototype for Zero-Day Attack Detection. *International Journal of Engineering and Advanced Technology*, 8(4), 1326–1329.
- Satyanarayana, E. A. G. (2023). Efficient Inductive Transfer Learning based Framework for Zero-Day Attack Detection. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11(9), 552–559.
- <https://doi.org/10.17762/ijritcc.v11i9.8843>
- Satyanarayana, G., & Chatrapathi, K. S. (2025). Enhancing intrusion detection in the BoT-IoT dataset through genetic algorithms-based feature extraction and ensemble machine learning approaches. In: *Lecture Notes in Networks and*

- Systems. Singapore: Springer Nature. (pp. 381–392).
https://doi.org/10.1007/978-981-96-3939-7_33
- Shao, S., McAleer, S., Yan, R., & Baldi, P. (2019). Highly accurate machine fault diagnosis using deep transfer learning. *IEEE Transactions on Industrial Informatics*, 15(4), 2446–2455.
<https://doi.org/10.1109/tii.2018.2864759>
- Singla, A., Bertino, E., & Verma, D. (2019). Overcoming the Lack of Labeled Data: Training Intrusion Detection Models Using Transfer Learning. In: *Proceedings of the 2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, June 12-15, 2019, Washington, DC. 69-74.
<https://doi.org/10.1109/SMARTCOMP.2019.00031>
- Song, H. M., Woo, J., & Kim, H. K. (2020). In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21, 100198.
<https://doi.org/10.1016/j.vehcom.2019.100198>
- Thamilarasu, G., & Chawla, S. (2019). Towards Deep-Learning-Driven intrusion detection for the internet of things. *Sensors*, 19(9), 1977.
<https://doi.org/10.3390/s19091977>
- Wu, P., Guo, H., & Buckland, R. (2019). A transfer learning approach for network intrusion detection. *arXiv*. arXiv:1909.02352.
<https://doi.org/10.48550/arXiv.1909.02352>
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv*. arXiv:1411.1792.
<https://doi.org/10.48550/ARXIV.1411.1792>
- Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1), 43–76.
<https://doi.org/10.1109/jproc.2020.3004555>