



Article

Shared Autoencoder-Based Unified Intrusion Detection Across Heterogeneous Datasets for Binary and Multi-Class Classification Using a Hybrid CNN–DNN Model

Hesham Kamal * and Maggie Mashaly *

Networks Department, Faculty of Information Engineering and Technology (IET), German University in Cairo (GUC), New Cairo 11835, Egypt

* Correspondence: hesham.khalil@student.guc.edu.eg (H.K.); maggie.ezzat@guc.edu.eg (M.M.)

Abstract

As network environments become increasingly interconnected, ensuring robust cyber-security has become critical, particularly with the growing sophistication of modern cyber threats. Intrusion detection systems (IDSs) play a vital role in identifying and mitigating unauthorized or malicious activities; however, conventional machine learning-based IDSs often rely on handcrafted features and are limited in their ability to detect diverse attack types across disparate network domains. To address these limitations, this paper introduces a novel unified intrusion detection framework that implements “Structural Dualism” to integrate three heterogeneous benchmark datasets (CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23) into a harmonized, protocol-agnostic representation. The framework employs a shared autoencoder architecture with dataset-specific projection layers to learn a unified latent manifold. This 15-dimensional space captures the underlying semantics of attack patterns (e.g., volumetric vs. signaling) across multiple domains, while dataset-specific decoders preserve reconstruction fidelity through alternating multi-domain training. To identify complex micro-signatures within this manifold, the framework utilizes a synergistic hybrid convolutional neural network–deep neural network (CNN–DNN) classifier, where the CNN extracts spatial latent patterns and the DNN performs global classification across twenty-five distinct classes. Class imbalance is addressed through resampling strategies such as adaptive synthetic sampling (ADASYN) and edited nearest neighbors (ENN). Experimental results demonstrate remarkable performance, achieving 99.76% accuracy for binary classification and 99.54% accuracy for multi-class classification on the merged dataset, with strong generalization confirmed on individual datasets. These findings indicate that the shared autoencoder-based CNN–DNN framework, through its unique feature alignment and spatial extraction capabilities, significantly strengthens intrusion detection across diverse and heterogeneous environments.

Academic Editor: Simon Tjoa

Received: 29 December 2025

Revised: 9 February 2026

Accepted: 10 February 2026

Published: 22 February 2026

Keywords: autoencoder; CNN–DNN; classification; combined framework; deep learning; IDS

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The relentless growth of emerging technologies and data-centric models, such as the internet of things (IoT), big data, and cloud computing, coupled with the increasing reliance of everyday communications on interconnected services, has rendered networked

computing indispensable. This, in turn, amplifies the critical importance of robust network security, as any security vulnerability or malicious threat could potentially compromise the entire infrastructure [1]. While conventional security measures like firewalls and encryption remain foundational, they are increasingly limited as attackers continuously develop more sophisticated intrusion techniques [2]. Consequently, cybersecurity experts have underscored the vital need to design effective network intrusion detection systems (IDSs) to safeguard modern networks. IDS solutions are specifically developed to maintain data availability, confidentiality, and integrity within networked environments by blocking unauthorized access and protecting critical information and communication systems [3]. Crucially, an efficient IDS must be capable of identifying both known and novel cyber threats with high accuracy while simultaneously maintaining a low false alarm rate [4].

IDSs generally employ two primary techniques: misuse detection and anomaly detection. Misuse detection, also known as signature-based detection, functions as a traditional method by identifying threats through the comparison of network activity against a database of previously recorded attack patterns, a method lauded for its high detection accuracy and low false positive rate. However, the continuous expansion of networks and services means that attackers are constantly developing new and unknown threats, which renders signature-based models vulnerable to these evolving risks [5]. Consequently, to ensure comprehensive network security, intrusion detection mechanisms must be capable of intelligently identifying and preventing both known and previously unseen attacks. Anomaly detection addresses this critical requirement by establishing a baseline of normal behavior and flagging deviations, thereby providing the crucial ability to detect both existing and novel threats, though it often comes with a higher false alarm rate.

Limitations of existing autoencoder-based IDS approaches, such as their inability to handle heterogeneous datasets or align features across domains, restrict their effectiveness in detecting diverse network attacks. The proposed shared projection autoencoder addresses these challenges by integrating multiple datasets with varying feature dimensions, learning a unified latent space while maintaining dataset-specific reconstruction through projection layers and decoders. This enables cross-dataset feature alignment, interleaved multi-domain training, and generalization across a wide spectrum of attacks, making it highly effective for detecting both known and novel intrusions in complex network environments, as shown in Table 1.

Table 1. Comparison of the proposed shared autoencoder with existing autoencoder variants.

Feature/Property	Standard AE	Multi-Task AE	Domain Adaptation AE	Proposed Shared Projection AE
Input compatibility	Homogeneous	Homogeneous	Homogeneous	Heterogeneous (variable dimensions)
Architectural focus	Reconstruction	Parallel tasking	Domain invariance	Cross-domain feature alignment
Feature mapping	Direct	Direct	Statistical/Rule-based	Learnable projection (non-linear)
Latent space	Isolated	Task-specific	Aligned (source-target)	Universal unified manifold
Data integration	Single source	Multi-source (fixed)	Transfer-based	Interleaved multi-domain jointly
Attack generalization	Narrow	Task-specific	Source-to-target	Universal (wide-spectrum)
Handling mismatch	Not possible	Manual padding	Requires shared features	Automatic via projection layers

This study introduces an innovative IDS that unifies multiple heterogeneous benchmark datasets, CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23, into a harmonized, feature-enriched representation. The fundamental novelty of this work lies in the architectural “Structural Dualism” of the proposed shared autoencoder. Unlike prior IDS studies that rely on a single dataset or simple feature concatenation, our approach acts as a behavioral

translator, aligning statistically disparate features into a singular unified latent manifold. The proposed framework employs dataset-specific projection layers followed by a shared encoder that learns a 15-dimensional latent space, capturing the underlying semantics of attack patterns, such as volumetric stress versus signaling, across different protocols. To achieve high-fidelity classification within this latent space, we utilize a synergistic hybrid convolutional neural network–deep neural network (CNN–DNN) architecture. In this design, the CNN layers are specifically engineered to extract spatial “micro-signatures” from the latent vector, while the DNN performs global relational mapping. This integrated approach allows for robust cross-dataset generalization, effectively identifying twenty-five traffic classes within a single model, a capability fundamentally missing in conventional serial autoencoder (AE)-CNN pipelines. The following summarizes this study’s main contributions:

- Introducing a novel IDS framework that integrates three heterogeneous benchmark datasets (CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23) into a harmonized, protocol-agnostic representation using a shared autoencoder. This enables the detection of twenty-five traffic classes within a single model by aligning disparate feature spaces into a universal behavioral manifold.
- Developing a shared autoencoder architecture that implements “Structural Dualism” to combine heterogeneous datasets into a unified 15-dimensional latent space. This design captures generalizable intrusion patterns (global intelligence) while preserving dataset-specific reconstruction (local nuance) through alternating multi-domain training, ensuring robust feature harmonization across different network environments.
- Providing an enhanced hybrid CNN–DNN deep learning model specifically engineered to process the learned latent space. This synergistic design uses the CNN for spatial pattern extraction within the 15-dimensional latent vector and the DNN for robust global classification, demonstrating superior performance over traditional serial models. Class imbalance is mitigated using resampling techniques such as adaptive synthetic sampling (ADASYN) and edited nearest neighbors (ENN).
- Undertaking rigorous evaluation of the proposed model on the merged datasets and each dataset individually to establish its generalization capability. This highlights its superior performance over existing solutions by proving the model recognizes the underlying semantics of attack behaviors rather than superficial dataset-specific features.

The manuscript is organized as follows: Section 2 provides an extensive overview of the pertinent academic literature. Section 3 describes the methods used in this investigation. The empirical results are presented in Section 4, while a detailed analysis of the findings is given in Section 5. Section 6 examines the limitations of the proposed method. The research is concluded by highlighting the main insights and contributions in Section 7. Lastly, Section 8 suggests potential directions for further research.

2. Previous Work

This section offers a concentrated review of intrusion detection methodologies, tracing their evolution from conventional machine learning techniques to sophisticated single (S) deep learning models. It specifically highlights recent hybrid (H) deep learning approaches and cross-dataset and multi-domain IDS that aim to improve detection performance and adaptability for binary (B) and multi-class (M) classification, as depicted in Table 2.

2.1. Traditional Machine Learning for IDS

The recent literature [6–11] has extensively utilized traditional machine learning (ML) paradigms, such as support vector machines (SVM), random forests (RF), and extreme gradient boosting (XGBoost), to identify network intrusions. These methodologies demonstrate high performance metrics, with accuracies often exceeding 98% for both binary and multi-class classification across diverse datasets like IoTID20, ToN-IoT, and UNSW-NB15. These studies primarily focus on developing ML-based IDS for detecting anomalies and DoS attacks within the context of IoT networks. While these traditional models offer implementation simplicity and effective detection rates, they are inherently limited by a heavy reliance on handcrafted feature engineering. Consequently, such approaches often struggle with scalability in high-traffic IoT environments and lack the generalizability required to adapt to novel, zero-day attack vectors. These documented shortcomings necessitate a shift toward more sophisticated and automated frameworks that couple multi-stage feature selection with intelligent optimization to achieve a more resource-aware workflow.

2.2. Deep Learning-Based Intrusion Detection

Recent research [12–18] has increasingly focused on deep learning methods like CNNs, long short-term memory networks (LSTMs), and generative models for finding intrusions. These models have performed well in both binary and multi-class classification on several datasets, including NSL-KDD, BoT-IoT, IoT-23, TON_IoT, MQTT-IoT-IDS2020, and CICIDS2017. Deep learning approaches have the benefit of automatically pulling out complex features from raw network traffic, which helps them find smaller groups of data and handle class imbalance through methods like resampling or creating new data with generative adversarial networks (GANs). These methodologies have achieved accuracies ranging from 93.3% to 99.74%, significantly outperforming traditional machine learning in capturing spatial and temporal patterns. However, despite how well they work, these models have some common downsides. They face scalability issues in large networks, need a lot of computing power and resources, and have limited generalizability across different environments. Furthermore, some models are specifically designed for multi-class situations, which makes them less useful for simpler binary classification tasks. A notable advancement in this area is the use of convolutional neural networks for autonomous malware detection and feature profiling, as surveyed by [19]. Beyond packet analysis, deep learning (DL) now tackles complex tasks like encrypted traffic fingerprinting and side-channel threats. Encryption-based identification has seen significant progress, as [20] successfully identified Android app identities within encrypted flows, while [21] utilized DL to eavesdrop on app activities via RF energy patterns. However, high computational costs remain a major barrier for real-time use on resource-constrained IoT devices.

2.3. Hybrid Models for IDS

The recent literature [22–35] has extensively explored hybrid deep learning models, integrating architectures such as CNN, LSTM, autoencoder, transformer, and multilayer perceptron (MLP) to enhance detection accuracy and system robustness. These hybrid frameworks have achieved very high performance, with reported accuracies reaching up to 99.99% on benchmark datasets such as IoT-23 and CSE-CIC-IDS2018. By combining the advantages of different models, utilizing CNNs for spatial feature extraction, LSTMs and gated recurrent units (GRUs) for temporal patterns, and Transformers for modeling long-range dependencies, these approaches effectively mitigate the limitations of standalone architectures. Furthermore, several of these studies incorporate GANs and optimization techniques to address persistent challenges like class imbalance and high false alarm rates.

Recent research highlights the shift toward specialized intelligence, where [36] surveyed Hybrid-CNN and large language model (LLM) integration for large-scale IoT data, and [37] explored LLM-assisted analysis for complex vulnerabilities. However, high computational costs and limited generalizability remain significant barriers to efficient, scalable deployment.

2.4. Cross-Dataset and Multi-Domain IDS

Recent work [38–46] has explored cross-domain and multi-view architectures to overcome the limitations of single-dataset models. These studies highlight that performance drops across datasets are often caused by feature naming mismatches and differing traffic distributions; for instance, uniform relabeling has been proven to restore accuracy across diverse datasets. Advanced frameworks in this group utilize Federated Learning for privacy-preserving multi-view fusion and Knowledge Graphs to capture complex relational features. Furthermore, deep stacked ensembles with “frozen” learners have been employed to preserve domain-specific knowledge, while principal component analysis (PCA)–Transformers have successfully recognized diverse traffic types across multiple aggregated data sources. Maintaining network reliability is vital; for instance, ref. [47] specifically improved secure, energy-efficient long range (LoRa) updates. However, multi-view processing still faces high overhead and latency. Research is now shifting toward joint representation learning to identify behavioral invariants across IoT, Cloud, and Enterprise environments, minimizing reliance on dataset-specific artifacts.

Table 2. Previous works.

Author	Dataset	Year	Utilized Technique	Accuracy		Contribution	Limitations
				B	M		
Esra Altulaihan. et al. [6]	IoTID20	2024	SVM	-	99.38%	This research aims to develop an ML-based IDS for detecting DoS attacks through anomaly detection in the context of IoT networks.	- Scalability - Generalizability - Traditional ML Models
Dainan Zhang et al. [7]	TON IoT, UNSW-NB15	2026	LightGBM	97.22 %	96.08%	First work to couple an SCC–SSA two-stage selection pipeline with Bayesian optimization for a fully automated, resource-aware IoT workflow	- Edge Testing - Dynamic Data
Abdallah R. Gad et al. [8]	ToN-IoT	2022	XGBoost	98.2%	97.8%	A machine learning-driven, decentralized system was introduced with the goal of identifying and decreasing IoT attacks	- Traditional ML Models - Class imbalance - Generalizability - Scalability
Nada Abdu Al-sharif et al. [9]	NSL-KDD	2023	RF	99.9%	-	An ML-driven IDS was developed in this study, with blockchain utilized to secure IoT device communications.	- Generalizability - Scalability - Traditional ML Models

Hanadi Hakami et al. [10]	UNSW-NB15	2025	RF	-	98.85%	This study introduces a new method that uses machine learning to boost detection accuracy and enhance system reliability.	- Scalability - Traditional ML Models
Mohamed ElKashlan et al. [11]	Iot-23	2023	Filtered classifier	99.2%	99.2%	This paper introduces a machine learning-based method to categorize and detect harmful traffic in IoT networks.	- Generalizability - Scalability - Traditional ML Models
Ahmed Abdelkhalik and Maggie Mashaly [12]	NSL-KDD	2023	CNN	93.3%	81.8%	The class imbalance in the NSL-KDD dataset is addressed in this research by using a CNN in conjunction with resampling approaches, which improves the detection of minority assaults.	Generalizability
Ajdani, M et al. [13]	NSL-KDD, CICSIDS2017	2025	GAN + DNN	98.2%	97.8%	Integrates GANs with DNNs to generate synthetic data, specifically targeting the reduction of False Positives and False Negatives.	- Resource Heavy - Adaptability
Tanzila Saba et al. [14]	BoT-IoT	2022	CNN	-	95.55%	In order to efficiently analyze all network traffic in IoT contexts, this study suggests a CNN-based approach for anomaly identification in IDS.	- Scalability - Generalizability - Class imbalance
Rubayyi Alghamdi and Martine Bellaiche [15]	Iot-23	2023	LSTM	98.20%	92.8%	In order to classify traffic into binary and multi-class categories, this work presents a deep ensemble IDS that uses Lambda architecture and LSTM-based models.	Resource use and computational complexity
Waad H. Aljuaid and Sultan S. Alshamrani [16]	CSE-CICSIDS2018	2024	CNN	-	98.67%	In order to increase the effectiveness of cyberattack detection in cloud computing environments, this study suggests a deep learning model that makes use of an advanced CNN architecture.	- Scalability - Absence of real-time assessment
Muhammad Wasim Nawaz et al. [17]	CICIDS2017	2023	LSTM	-	99%	In this study, an LSTM-based model with tailored loss functions for multi-class classification and oversampling is proposed to address class imbalance in network intrusion detection.	- Computational Complexity - Only applicable to issues involving numerous classes.
Chiming Xi et al. [18]	NSL-KDD, CIC-DDoS 2019, UNSW-NB15	2024	Multi-Scale Transformer (IDS-MTran)	99.25%	99.74%	Introduced Multi-Scale branches using convolution kernels to capture both micro-details and macro-patterns. Features the cross feature enrichment (CFE) module for deep fusion.	- Class imbalance - Hardware Requirements
Hesham Kamal and Maggie Mashaly [22]	CSE-CICIDS2018	2025	AE-DTNN	99.92%	99.72%	This paper details a hybrid AE-DTNN model engineered to produce a highly dependable and agile IDS for deployment in current, dynamic network ecosystems.	- Scalability - Generalizability

Yanfang Fu et al. [23]	NSL-KDD	2022	CNN and BiLSTMs	90.73 %	-	In order to improve network intrusion detection accuracy and robustness, this research presents DLNID, a deep learning model that makes use of CNN, attention, and Bi-LSTM.	•	Absence of real-time assessment
Emad Ul Haq Qazi et al. [24]	CSE-CIC-IDS2018	2023	CNN-RNN	99.02 %	98.90%	Developed a dual-stage fusion model. The CNN layer captures local spatial features (relationships between feature columns), while the RNN layer extracts temporal patterns (sequential dependencies in traffic flows) within the CICIDS-2018 dataset.	•	Generalizability
Muhammad Sajid et al. [25]	CICIDS2017	2024	CNN-LSTM	97.90 %	-	In order to enhance the detection of emerging network assaults, this study proposes a hybrid model that combines XGBoost, CNN, and LSTM.	•	scalability
Hesham Kamal and Maggie Mashaly [26]	Iot-23	2025	CNN-MLP	99.99 %	99.91%	A hybrid model, which merges a CNN and an MLP, is introduced in this study to accurately spot and sort IoT traffic in both dual and multi-category contexts.	•	class imbalance
Muhammad Basit Umair et al. [27]	NSL-KDD	2022	Multilayer CNN-LSTM	-	99.5%	An IDS utilizing CNN and LSTM with a softmax classifier is presented in this study. It is assessed on benchmark datasets and contrasted with a multilayer DNN.	•	Generalizability
Hesham Kamal and Maggie Mashaly [28]	NF-BoT-IoT-v2	2025	Transformer-DNN & AE-CNN	99.98 %	97.90% & 97.95%	They propose two sophisticated hybrid deep learning models: the first integrates an autoencoder with a CNN, while the second merges a Transformer with a DNN.	•	Generalizability
Sami Yaras and Murat Dener [29]	ToN-IoT	2024	CNN-LSTM	98.75 %	-	In order to increase intrusion detection accuracy, this study created a CNN-LSTM model using pertinent characteristics from the CICIoT2023 and TON_IoT datasets.	•	Scalability
Hesham Kamal and Maggie Mashaly [30]	NF-UNSW-NB15-v2	2024	Transformer-CNN	99.71 %	99.02%	The study proposes a combined Transformer-CNN model designed to solve the problem of class imbalance by applying resampling and class weighting techniques.	•	Scalability
Zhi-Xian Zheng [31]	ToN IoT	2025	GAN + Transformer + Bi-GRU	99.9%	97.66%	Combines GANs to generate adversarial samples for class balancing with Transformer/Bi-GRU for global and sequential feature extraction.	•	Generalizability
Mr. S. Balaji et al. [32]	IoT-specific Attack Data	2024	Hybrid GAN + Firefly Optimization	99%	98%	Developed a Distributed IDS using Firefly optimization and SMOTE to detect malicious behavior without a centralized controller.	•	Recall Variability
Doaa Mohsin Abd Ali Afraji [33]	ToN_IoT, CICIDS2017	2025	CNN-LSTM-GRU (Parallel-Sequential Fusion)	99.99 %	99.49%	Introduced a multi-branch architecture that captures spatial (CNN) and both short-term (GRU) and long-term (LSTM) temporal patterns simultaneously.	•	Computational Cost
Kuburat Oyeranti Adefemi [34]	IoTID20, BoT-IoT	2025	Hybrid CNN-GRU	99.83 %	99.01%	Integrated convolutional layers for spatial feature extraction with GRUs for temporal dependencies, specifically	•	Edge Deployment
							•	Interpretability

						choosing GRU over LSTM for lower computational overhead.		
Md Aadil Hasan and Dev Sharma [35]	CIC-IDS2017, UNSW-NB15, WSN-DS	2025	Hybrid CNN-LSTM (3-Layer)	99.65 %	99.61%	Combines CNN for spatial pattern extraction with LSTM for temporal dependency modeling; uses Stratified K-Fold (K = 8) for validation stability.	Class Imbalance	
Minxiao Wang et al. [38]	CIC-IDS-2017, UNSW-15, and ISCX-2012	2024	CNN	97%	96.2%	An ensemble IoT IDS was created by researchers, combining ML and DL techniques to boost its ability to detect attacks.	- Class imbalance - Generalizability - Absence of Real-Time Assessment	
Said Al-Riyami et al. [39]	NSL-KDD, FigureKDD	2021	Trees, k-NN, CNN, LSTM	96.51 %	89.80%	Identified that cross-dataset failure is caused by feature naming mismatches; proved uniform relabeling restores performance.	- Class imbalance - Generalizability - Absence of Real-Time Assessment	
Jia Yu et al. [40]	TON_IoT, UNSW-NB15, Synthetic (GAN)	2025	CAE-NSVM (Multi-view Fusion + Federated Learning)	96.3%	82.9%	Integrated AE-SVM joint loss model that extracts features from 5 host views within a privacy-preserving Federated framework.	- Class imbalance - Generalizability - Absence of Real-Time Assessment - Computational Overhead of “Multi-View” Processing	
Adel Alabbadi and Fuad Bajaber [41]	TON_IoT, NSL-KDD, CICIoMT	2025	X-FuseRLSTM (Sparse Transformer + RLSTM + XAI)	99.72 %	99.40%	Dual-path fusion using Transformers for long-range spatial data and Residual LSTMs for temporal patterns, integrated with XAI (SHAP/LIME).	- Interpretability Gap - Complexity	
Muhammad Iqbal Amin et al. [42]	NFv2-(UNSW-NB15, BoT-IoT, ToN-IoT, CIC-2018)	2026	Heterogeneous Deep Stacked Ensemble (GRU + LSTM + DNN + MLP)	99.83 %	88.77%	Uses “Frozen” heterogeneous base learners to preserve domain-specific knowledge and a Meta-Learner to bridge the gap between different network traffic distributions.	- High Resource Demand - Dataset Specificity	
Min Li et al. [43]	TON_IoT, UNSW-NB15	2025	Multi-View KG-Enhanced DL (Model 6: KG + Multi-CNN + LSTM)	97.3%	91.1%	Proposed a two-level fusion (secondary fusion) strategy integrating Knowledge Graphs for relational features with DL for spatial/temporal data.	- High computational overhead - Latency	
Hesham Kamal and Maggie Mashaly [44]	CICIDS2017 and CSE-CIC-IDS2018)	2025	PCA-Transformer	99.80 %	99.28%	A modern IDS design is presented that uses an aggregated dataset and a PCA-Transformer technique to recognize	Dataset Combination on	

						twenty-one types of traffic (1 benign and 20 malicious) drawn from multiple data sources.		- different datasets - Generalizability
Rubayyi Alghamdi and Martine Bellaiche [45]	UNSW-NB15, Ton_IoT and IoT-23	2022	RF	99.45 %	97.81%	An ensemble IoT IDS was created by researchers, combining ML and DL techniques to boost its ability to detect attacks.	- Class imbalance - Generalizability - Scalability - Traditional ML Models	
Saleh Alabdulwahab et al. [46]	TON_IoT, BoT-IoT and MQTT-IoT-IDS2020	2023	CTGAN	-	99%	This technique uses CTGAN to generate fake IoT intrusion reports, which improves detection performance and guarantees data balance.	Generalizability	

2.5. Challenges

The effectiveness and practical implementation of deep learning-driven intrusion detection systems are limited by a number of challenges. Combining datasets to identify a variety of attack types, complex combination architecture, attaining high performance, guaranteeing generalization, and preserving scalability are some of these difficulties. The main issues are listed below.

- **Combination of Datasets for Detecting a Broader Range of Attack Types:** A significant challenge is that individual datasets often contain only a limited variety of attack types, which can hinder the effectiveness of intrusion detection models. The root cause of this issue is the heterogeneity of networks, where devices generate diverse traffic and face varied attack scenarios.
- **Complexity and Difficulty of Combined Architectures:** A major challenge lies in designing methods that can effectively integrate multiple heterogeneous datasets while preserving their unique characteristics. The root cause is that traditional approaches require numerous preprocessing and integration steps, as well as careful design of model architectures.
- **High Performance:** A key challenge in intrusion detection is achieving consistently high performance across different environments and attack types. The root cause of this challenge is the complexity and diversity of network traffic, where devices generate large volumes of heterogeneous data with varying patterns, making it difficult for models to maintain accuracy, precision, and reliability under all conditions.
- **Class Imbalance:** A major challenge in intrusion detection is that datasets often contain a disproportionate number of samples for different attack types, with some attacks being significantly underrepresented. The root cause of this issue is the infrequent occurrence of certain attack types in real-world networks, which leads to skewed data distributions and can reduce model performance for minority classes.
- **Generalization:** A key problem in intrusion detection is that models trained on specific datasets often perform poorly when faced with new or different environments. The root cause of this issue is the heterogeneity of networks and traffic patterns, where devices, protocols, and attack behaviors vary widely, making it difficult for models to generalize beyond the data they were trained on.
- **Scalability:** A main problem with intrusion detection is keeping it effective as the size of networks and the amount of traffic grows. The root cause of this issue is the rapid

growth and expansion of ecosystems, where a large number of heterogeneous devices generate massive amounts of data, making it difficult for models to process and analyze all traffic efficiently.

The benefits and limitations of other models in relation to our CNN–DNN method are understated in Table 3, which compares comparable research with our methodology.

Table 3. Comparison of Previous Works with Our Work.

Author	High Performance		Classification Type		Class Imbalance Mitigation	Generalization Threshold > 2	Scalability	Learning Type		Model Type		Datasets Combination
	B	M	B	M				DL	ML	H	S	
Esra Altulaihan. et al. [6]		✓		✓					✓		✓	
Saleh Alabdulwahab et al. [7]	✓		✓	✓					✓		✓	
Abdallah R. Gad et al. [8]	✓	✓	✓	✓	✓				✓		✓	
Nada Abdu Al-sharif et al. [9]	✓		✓						✓		✓	
Hanadi Hakami et al. [10]		✓		✓	✓	✓			✓		✓	
Mohamed ElKashlan et al. [11]	✓	✓	✓	✓					✓		✓	
Ahmed Abdelkhalek and Maggie Mashaly [12]			✓	✓	✓			✓			✓	
Ajdani, M et al. [13]	✓	✓	✓	✓	✓			✓			✓	
Tanzila Saba et al. [14]				✓				✓			✓	
Rubayyi Alghamdi and Martine Bellaiche [15]	✓	✓	✓	✓				✓			✓	
Waad H. Aljuaid and Sultan S. Alshamrani [16]	✓	✓		✓	✓	✓		✓			✓	
Muhammad Wasim Nawaz et al. [17]		✓		✓		✓		✓			✓	
Chiming Xi [18]	✓	✓	✓	✓	✓	✓		✓			✓	
Hesham Kamal and Maggie Mashaly [22]	✓	✓	✓	✓	✓	✓		✓			✓	
Yanfang Fu et al. [23]			✓		✓			✓			✓	
Emad Ul Haq Qazi et al. [24]	✓	✓	✓	✓				✓			✓	
Muhammad Sajid et al. [25]	✓		✓		✓			✓			✓	
Hesham Kamal and Maggie Mashaly [26]	✓	✓	✓	✓	✓	✓		✓			✓	
Muhammad Basit Umair et al. [27]		✓		✓				✓			✓	
Hesham Kamal and Maggie Mashaly [28]	✓	✓	✓	✓	✓			✓			✓	
Sami Yaras and Murat Dener [29]	✓		✓					✓			✓	
Hesham Kamal and Maggie Mashaly [30]	✓	✓	✓	✓	✓			✓			✓	
Zhi-Xian Zheng [31]	✓	✓	✓	✓	✓			✓			✓	
Mr. S. Balaji et al. [32]	✓	✓	✓	✓	✓			✓	✓		✓	

Doaa Mohsin Abd Ali Afraji [33]	✓	✓	✓	✓	✓			✓	✓
Kuburat Oyeranti Adefemi [34]	✓	✓	✓	✓	✓			✓	✓
Md Aadil Hasan and Dev Sharma [35]	✓	✓	✓	✓	✓			✓	✓
Minxiao Wang et al. [38]	✓		✓	✓		✓		✓	✓
Said Al-Riyami et al. [39]			✓	✓				✓	✓
Jia Yu et al. [40]			✓	✓		✓		✓	✓
Adel Alabbadi and Fuad Bajaber [41]	✓	✓	✓	✓	✓	✓		✓	✓
Muhammad Iqarar Amin [42]	✓		✓	✓	✓	✓		✓	✓
Min Li et al. [43]	✓	✓	✓	✓				✓	✓
Hesham Kamal and Maggie Mashaly [44]	✓	✓	✓	✓	✓	✓		✓	✓
Rubayyi Alghamdi and Martine Bellaiche [45]	✓	✓	✓	✓				✓	✓
Saleh Alabdulwahab et al. [46]		✓		✓				✓	✓
Our Work	✓	✓	✓	✓	✓	✓	✓	✓	✓

The proposed approach successfully overcomes these obstacles by improving its capacity to identify various attack types, complex combination architecture, attaining better performance, preserving scalability, and improving generalization. These problems have solutions, which are listed below.

- **Combination of Datasets for Detecting a Broader Range of Attack Types:** This work introduces a novel IDS framework that integrates the heterogeneous CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23 datasets into a unified, feature-enriched representation using a shared autoencoder. This integration enables a single model to detect a broad spectrum of threats across twenty-five traffic classes, comprising one benign class and twenty-four distinct attack types.
- **Complexity and Difficulty of Combined Architectures:** The proposed approach employs a shared encoder with dataset-specific decoders, enabling the model to learn generalizable features across diverse traffic patterns while preserving all information and maintaining reconstruction fidelity and detection performance. This shared autoencoder framework is flexible, can accommodate any number of heterogeneous datasets, and follows the same integration steps.
- **High Performance:** High detection performance is achieved using the proposed hybrid CNN–DNN model with enhanced preprocessing, leading to superior results in both binary and multi-class classification tasks.
- **Class Imbalance:** Effectively addressed through mitigation strategies such as ADASYN, and ENN, ensuring balanced learning and improved detection performance.
- **Generalization:** Ensured through training on both the unified multi-dataset representation (CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23) and the individual datasets, resulting in improved robustness and adaptability to varied attack environments
- **Scalability:** Achieved through an optimized hybrid CNN–DNN architecture designed to manage large-scale traffic derived from both the unified representation of

CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23 and from each dataset independently, while preserving strong detection performance.

3. Methodology

Existing works show that intrusion detection systems based on machine and deep learning face numerous challenges that limit both their efficiency and their applicability in real-world scenarios. While components like autoencoders and CNNs are common in the IDS literature, they are typically implemented as isolated, single-domain solutions. These difficulties include challenges in merging multiple datasets to encompass a broad spectrum of attack behaviors, maintaining consistently high accuracy across diverse operating conditions, and limited adaptability when exposed to unfamiliar traffic distributions. Table 4 compares independent encoders, shared encoders, and our proposed Shared autoencoder. The comparison highlights the trade-offs between generalization across datasets and retention of dataset-specific information, motivating the need for a hybrid approach that maximizes both.

Table 4. Autoencoder architectures: generalization vs. information retention.

Architecture	Mechanism	Generalization	Information Retention
Independent encoders	Separate AE per dataset	Low	High locally
Shared encoder <i>w/o</i> vertical combination	Single AE, shared latent	High	Low; unique attack patterns lost
Proposed method	Shared AE + dataset-specific latent & decoders	Maximum	Max, keeps universal & unique patterns

To overcome these issues, we propose a novel IDS framework that integrates the heterogeneous CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23 datasets into a unified, feature-enriched representation. The fundamental novelty of this design lies in its “Structural Dualism”: it acts as a behavioral translator that aligns statistically disparate features (e.g., PCAP-based timing vs. Netflow volumes) into a singular unified latent manifold. The learned 15-dimensional latent features are then processed by a hybrid CNN–DNN classifier, which is specifically engineered for this latent space. In this synergistic design, the CNN layers function as spatial feature extractors to identify local correlations and “micro-signatures” within the latent vector, while the DNN layers perform global relational mapping to achieve the final classification. This hierarchical approach allows the model to capture both universal intrusion patterns and unique dataset-specific characteristics without requiring domain labels, a capability fundamentally missing in conventional serial AE–CNN pipelines. Challenges related to skewed class distributions are mitigated through advanced data balancing techniques, such as ADASYN and ENN. Generalization is further strengthened by training on both the unified multi-dataset representation and each dataset individually, ensuring robust adaptability across diverse attack environments. Scalability is maintained through the lightweight yet efficient CNN–DNN design, which handles large-scale network traffic without compromising detection performance. The workflow of the proposed system architecture is depicted in Figure 1, which illustrates the generalized architectural framework; specific preprocessing modules are applied conditionally based on the unique characteristics of each individual dataset as detailed in the following sections.

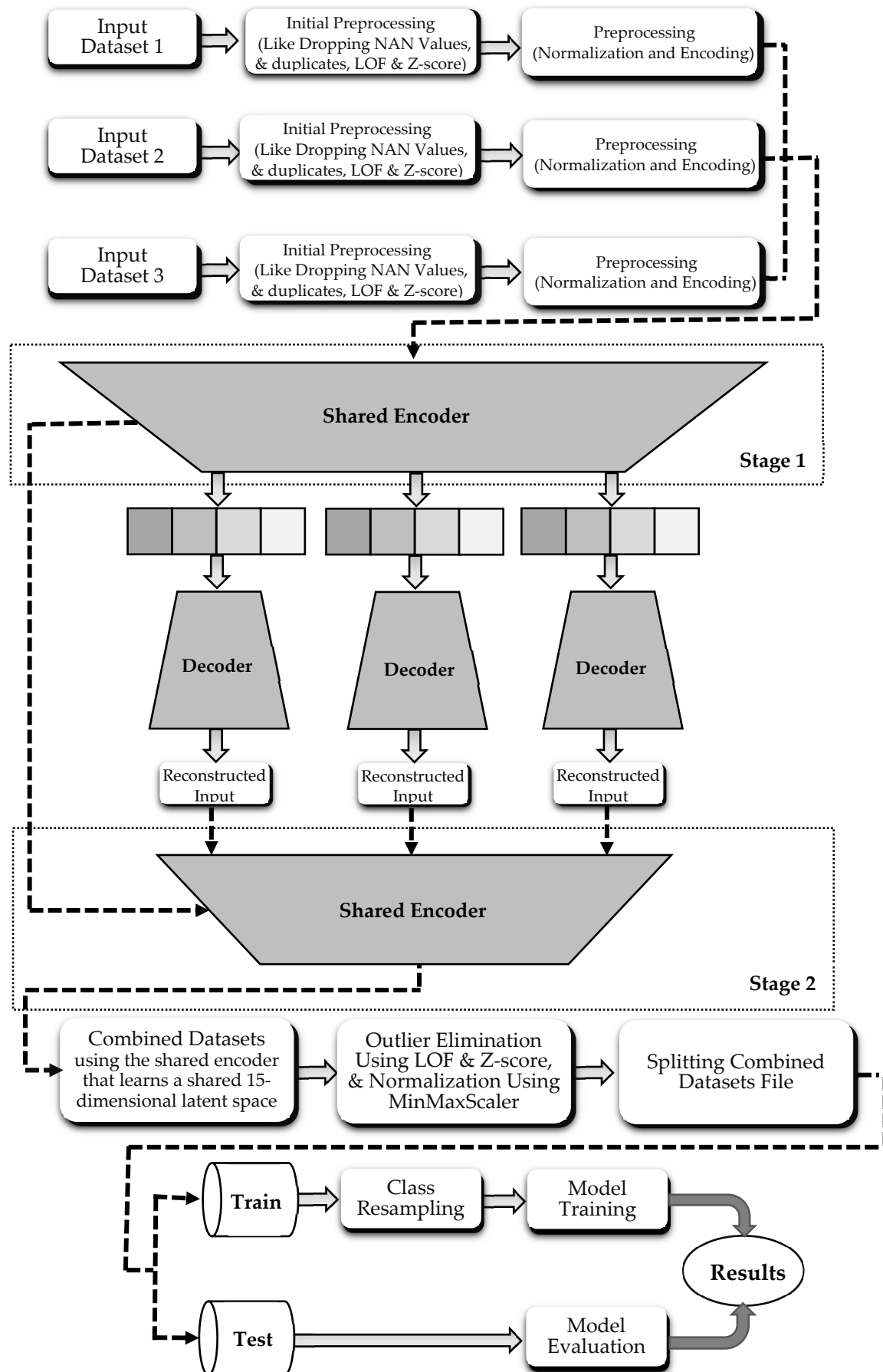


Figure 1. Workflow of the proposed system architecture using the combined dataset.

3.1. Description of Dataset

The proposed intrusion detection framework was evaluated on a unified dataset combining CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23, covering diverse network traffic with both benign and malicious activities for binary and multi-class classification. The CSE-CIC-IDS2018 dataset [48,49], created by the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC), replicates realistic network traffic and contains 80 features across 15 classes, including one benign and fourteen malicious categories. NF-BoT-IoT-v2 [50,51], based on NetFlow records from IoT devices, comprises over 37 million flows, with 0.36% benign and the remainder malicious, organized into five classes. The IoT-23 dataset [11,26,52–55], developed by Stratosphere Lab with support from Avast, includes 23 scenarios of IoT traffic, with three benign and 20 malicious scenarios labeled by malware family, providing rich traffic patterns suitable for both binary and multi-class tasks.

3.2. Shared Autoencoder Framework

The preprocessing of the datasets involved steps to ensure data quality and suitability for model training. For CSE-CIC-IDS2018, the ten daily CSV files were consolidated, duplicates were removed, representative sampling was performed, and outliers were detected and eliminated using local outlier factor (LOF) and Z-score methods. The NF-BoT-IoT-v2 dataset underwent sampling, followed by outlier detection and elimination using Z-score and LOF techniques. Similarly, the IoT-23 dataset was preprocessed by sampling, removal of records containing missing values, removing duplicate records, and eliminating extreme outliers using the Z-score method to enhance model performance. The CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23 datasets were merged into a single unified dataset using a shared autoencoder. The framework employs dataset-specific projection layers followed by a common encoder that learns a shared 15-dimensional latent space to capture generalizable intrusion patterns across all datasets. Dataset-specific decoders preserve reconstruction accuracy through alternating multi-domain training, while vertical combination is applied to integrate latent space features from multiple datasets, improving the model's ability to learn cross-domain representations without losing dataset-specific information. This design enables the shared autoencoder-based CNN–DNN framework to detect a total of 25 traffic classes, including one benign and 24 attack types. After merging, extreme values are identified and removed using the LOF and Z-score method to prevent negative impacts on model performance. Continuous features are normalized via MinMaxScaler. Finally, training and testing subsets are created from the combined dataset, then class imbalance is mitigated on the training set using resampling strategies such as ADASYN and ENN, and the model's performance is thoroughly evaluated.

1. Dataset Preparation and Feature–Label Separation

In this work, three heterogeneous intrusion detection datasets, CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23, are utilized to construct a unified representation learning framework. Each dataset contains a distinct set of features and labeling schemes. To ensure unsupervised representation learning, label-related attributes are removed from the feature space prior to model training. Specifically, the columns Label, Attack, and detailed_label are separated as target variables, while the remaining attributes form the input feature matrices for each dataset. This step ensures that the autoencoder learns intrinsic data structures without supervision or label leakage. The theoretical rationale for this strict separation is to allow the shared autoencoder to function as a protocol-agnostic behavioral translator, mapping raw features into a latent space where labels are only reintroduced during the final classification stage to ensure cross-domain generalization. The feature matrices and labels for each dataset are defined in Equations (1) and (2) [56].

$$X^{(d)} = \frac{\text{Features of dataset } d}{(\text{Label column})} \quad (1)$$

$$y^{(d)} = \text{Label column of dataset} \quad (2)$$

where $X^{(d)}$ is the input feature matrix of dataset $d \in \{CSE - CIC - IDS2018, NF - BoT - IoT - v2, IoT - 23\}$, containing all relevant features for unsupervised learning, and $y^{(d)}$ is the corresponding label vector used for downstream classification tasks.

2. Feature Encoding and Missing Value Handling

After feature–label separation, each dataset is preprocessed independently to ensure numerical consistency and robustness during neural network training. Missing values are replaced with zeros to prevent undefined operations during forward propagation. Categorical features, which cannot be directly processed by neural networks, are converted into numerical form using label encoding, as shown in Equation (3) [57,58], where each unique categorical value is mapped to an integer index. This independent encoding strategy is theoretically motivated by the need to preserve the domain-specific semantics of each dataset. By processing each dataset separately before the shared projection stage, the framework ensures that categorical mappings, such as protocol flags or service types, do not suffer from index collision, where identical numerical values could otherwise represent fundamentally different behaviors across heterogeneous network environments. Following this transformation, all features are cast to 32-bit floating-point format to ensure computational efficiency and compatibility with TensorFlow operations.

$$X_{i,j}^{(d)} = \begin{cases} 0 & \text{if missing} \\ \text{LabelEncode}X_{i,j}^{(d)} & \text{if categorical} \\ X_{i,j}^{(d)} & \text{Otherwise} \end{cases} \quad (3)$$

where $X_{i,j}^{(d)}$ denotes the value of the j -th feature for the i -th sample in dataset d .

3. Dataset-Specific Feature Normalization

Due to differences in scale and statistical distribution across datasets, Min–Max normalization is applied independently to each dataset. For each feature, values are scaled into the range $[0, 1]$, as shown in Equation (4) [57,59–61], ensuring numerical stability and preventing features with larger magnitudes from dominating the learning process. This independent normalization strategy is theoretically critical as the primary stage of Manifold Alignment. By scaling each dataset within its own statistical bounds before the data enters the shared architecture, the framework ensures that the shared autoencoder receives inputs on a uniform scale. This prevents datasets with higher-magnitude features from disproportionately biasing the shared weights of the behavioral translator, thereby maintaining the integrity and fairness of the unified latent space.

$$N_{i,j}^{(d)} = \frac{X_{i,j}^{(d)} - \min(X_{i,j}^{(d)})}{\max(X_{i,j}^{(d)}) - \min(X_{i,j}^{(d)})} \quad (4)$$

where $X_{i,j}^{(d)}$ is the j -th column of dataset d .

4. Definition of Model Hyperparameters

Following preprocessing, the dimensionality of each dataset's feature space is determined dynamically from the processed input matrices. Two key hyperparameters are defined: a projection dimension of 32 neurons to align heterogeneous feature spaces and a latent dimension of 15 neurons to represent compact, shared embeddings. The selection of a 15-dimensional latent space is theoretically motivated by the principle of Intrinsic Dimensionality. While raw network traffic features are high-dimensional, ranging from

15 to 67 features in this study, they exhibit high mutual information and redundancy. A 15-dimensional bottleneck was identified via empirical sensitivity analysis as the optimal compression knee where the reconstruction error stabilizes. This specific dimensionality serves as a non-linear information filter that suppresses dataset-specific statistical noise while forcing the Shared Encoder to prioritize universal behavioral signatures, such as volumetric stress and protocol signaling patterns, that are invariant across different network do-mains. This prevents the curse of dimensionality and ensures the downstream CNN–DNN classifier operates on a high-density semantic manifold. These values are selected to balance representational capacity and dimensionality reduction while avoiding overfitting. To ensure mathematical clarity throughout the subsequent sections, the formal notation for these data transformation stages is summarized in Table 5.

Table 5. Notation for feature transformation.

Term	Symbol	Description
Original features	X_d	Raw input matrix after label separation.
Projected features	P_d	Aligned features via projection layers.
Latent features	Z_d	Embeddings extracted by shared encoder.
Unified features	$Z_{Combined}$	Vertically concatenated latent matrices.

5. Input Layer Construction

To accommodate datasets with different feature dimensionalities, three independent input layers are defined, each corresponding to one dataset. These input layers receive normalized feature vectors from their respective datasets and serve as entry points to the neural architecture. This architectural choice is theoretically motivated by the need for independent domain entry points; it ensures that the unique structural integrity of each heterogeneous dataset is preserved before they are unified in the shared latent space. By maintaining separate entry points, the framework avoids the information loss or bias that typically occurs when forcing diverse feature sets into a single, static global schema. This design enables parallel processing while maintaining dataset-specific input handling.

6. Dataset-Specific Projection Layers

Given the heterogeneous nature of the input feature spaces, each dataset is first passed through a dataset-specific projection layer implemented as a fully connected dense layer with 32 neurons and ReLU activation. These layers function as a “Dimensional Alignment Bridge.” Their theoretical purpose is to map raw features of varying dimensions into a common intermediate space ($p = 32$), allowing the subsequent shared encoder to treat all datasets as semantically equivalent. This prevents the model from being biased toward datasets with higher feature counts and ensures that the shared weights are optimized based on behavioral patterns rather than input density. By creating this uniform buffer, the framework achieves structural dualism, where dataset-specific nuances are translated into a standardized format before deep feature extraction. By performing this transformation, all datasets are aligned into a shared representational space, enabling effective parameter sharing in subsequent encoder layers without requiring manual feature harmonization, as shown in Equation (5) [62–64].

$$H^{(d)} = f_{\text{proj}}^{(d)}(N^{(d)}) = \sigma(W_{\text{proj}}^{(d)} N^{(d)} + b_{\text{proj}}^{(d)}) \quad (5)$$

where σ is ReLU activation, $W_{\text{proj}}^{(d)} \in R^{\text{features}_d \times P}$.

7. Shared Encoder Architecture

The core of the proposed framework is a shared encoder that processes the projected representations from all datasets using identical trainable parameters. The shared encoder

consists of two fully connected hidden layers with 64 and 32 neurons, respectively, using ReLU activation, followed by a latent layer with 15 neurons and linear activation. The theoretical motivation for weight sharing is to facilitate Cross-Domain Knowledge Transfer. By forcing the encoder to map three different input distributions into a single shared manifold, the model acts as a Behavioral Invariant Extractor. This architecture ensures that the learned embeddings prioritize the fundamental mechanics of network intrusions, such as the relationship between packet frequency and volumetric stress, over the specific formatting of any single dataset. This shared learning process allows the framework to generalize to unseen attack variants across heterogeneous environments. The low-dimensional latent space ($L = 15$) helps suppress noise, reduce redundancy, and improve generalization capability. The shared encoder maps the projected feature representation of each dataset into a latent space. For a given dataset d , the encoder produces a latent vector $Z^{(d)} \in H^L$, where L denotes the latent dimension. The transformation is defined in Equations (6)–(8) [65–67].

$$h_1^{(d)} = \sigma(W_1 H^{(d)} + b_1) \quad (6)$$

$$h_2^{(d)} = \sigma(W_2 h_1^{(d)} + b_2) \quad (7)$$

$$Z^{(d)} = W_L h_2^{(d)} + b_L \quad (8)$$

where W_1 , W_2 , W_L and b_1 , b_2 , b_L denote the weight matrices and bias vectors of the encoder layers, respectively, and ReLU is applied to the hidden layers, while linear activation is applied at the final latent layer to generate the latent representation.

8. Dataset-Specific Decoder Design

To reconstruct the original inputs, a separate decoder is assigned to each dataset. Each decoder progressively expands the shared latent representation back to the original feature dimensionality of its corresponding dataset through a sequence of fully connected layers. Specifically, the decoder consists of two hidden layers with 32 and 64 neurons, respectively, as shown in Equations (9)–(11) [57,63,65,68,69]. The use of dataset-specific decoders is theoretically motivated by the need to enforce the representational fidelity of the shared manifold. While the encoder focuses on finding universal commonalities, the individual decoders must prove that the 15-dimensional latent space contains sufficient information to recover the unique statistical nuances of each original domain. This asymmetric reconstruction approach ensures that the shared features are not just generic noise but are high-fidelity descriptors capable of representing diverse network environments. ReLU activations are used in hidden layers, while sigmoid activation is applied in the output layer to match the normalized feature range. This design ensures accurate reconstruction while preserving dataset-specific characteristics.

$$h_3^{(d)} = \sigma(W_3 Z^{(d)} + b_3) \quad (9)$$

$$h_4^{(d)} = \sigma(W_4 h_3^{(d)} + b_4) \quad (10)$$

$$N^{(d)} = \sigma_{Sigmoid}(W_5 h_4^{(d)} + b_5) \quad (11)$$

where W_3 , W_4 , W_5 and b_3 , b_4 , b_5 denote the weight matrices and bias vectors of the decoder layers, respectively, and σ represents the ReLU activation function. A sigmoid activation is applied at the output layer to match the normalized feature range of the input data.

9. Model Compilation and Optimization Strategy

The complete multi-input, multi-output autoencoder model is optimized using the Adam optimizer due to its adaptive learning rate and fast convergence properties. Mean

squared error (MSE) is employed as the reconstruction loss for each dataset, as shown in Equation (12) [64,70,71]. The total loss function is theoretically designed as a “Multi-Task Balancing Constraint.” By aggregating per-dataset reconstruction errors with equal weighting, the framework prevents any single dataset from dominating the manifold orientation of the shared encoder. This ensures that the 15-dimensional latent space remains a “Fair Representation” of all three network environments (IoT, Cloud, and Traditional IDS), rather than over-fitting to the statistical majority of a single data source. Masking mechanisms are applied during interleaved batch training to exclude zero-padded dummy samples, ensuring that only valid samples contribute to gradient updates.

$$L = \alpha \left\| O^{\text{CSE-CIC-IDS2018}} - R^{\text{CSE-CIC-IDS2018}} \right\|_2^2 + \beta \left\| O^{\text{NF-BoT-IoT-v2}} - R^{\text{NF-BoT-IoT-v2}} \right\|_2^2 + \gamma \left\| O^{\text{IoT-23}} - R^{\text{IoT-23}} \right\|_2^2 \quad (12)$$

where $O^{(d)}$ and $R^{(d)}$ denote the original and reconstructed feature matrices of dataset d , respectively, and the weighting coefficients satisfy $\alpha + \beta + \gamma = 1$. In practice, equal weighting ($\alpha = \beta = \gamma = 1/3$) is applied, and the masking ensures that inactive datasets in interleaved mini-batches do not influence the shared encoder or decoder updates. This optimization strategy allows the shared encoder to learn robust, aligned, and generalizable latent representations across heterogeneous datasets, while dataset-specific decoders preserve reconstruction fidelity by providing local feedback loops.

10. Interleaved and Masked Multi-Dataset Training Strategy

To mitigate dataset bias and enable balanced cross-domain representation learning, an interleaved and masked multi-dataset training strategy is employed. During training, mini-batches from the three datasets are processed in an interleaved manner within each epoch. When a dataset does not provide samples for a given batch position, its input is padded with zero-valued dummy vectors and excluded from optimization using a masking mechanism, as formulated in Equation (13) [64,72,73]. This strategy is theoretically motivated by the need to prevent Catastrophic Forgetting, which often occurs when a model is trained on multiple domains sequentially. By interleaving the data, the Shared Encoder is forced to maintain a consistent gradient direction that satisfies all three domains simultaneously. Furthermore, the masking mechanism acts as a “Gradient Gate,” ensuring that the specific projection and decoder weights for an inactive dataset are not corrupted by noise from the other active data streams. This ensures that only active samples contribute to the reconstruction loss, while inactive pathways do not influence parameter updates.

$$h_{\text{dummy}}^{(D)} = 0, \forall D \neq d \quad (13)$$

To ensure reproducibility, the detailed logic for this interleaved masked training is provided in Algorithm 1.

Algorithm 1: Interleaved and Masked Multi-Dataset Training

Input: Multi-dataset feature matrices and training hyperparameters.

Output: Trained shared autoencoder and unified latent space representations.

- 1: Normalize each dataset independently.
 - 2: Split each dataset into training and validation sets.
 - 3: Initialize model parameters (projection, encoder, decoders).
 - 4: for epoch = 1 to E do
 - 5: Generate interleaved mini-batches from all datasets
 - 6: for each interleaved batch do
 - 7: Project inputs using dataset-specific projection layers
 - 8: Encode projected features using shared encoder
 - 9: Reconstruct inputs using corresponding dataset-specific decoders
 - 10: Compute masked reconstruction loss for each dataset
-

-
- 11: Aggregate total loss across datasets
 - 12: Update θ_{proj} , θ_{enc} , θ_{dec} using backpropagation
 - 13: end for
 - 14: Evaluate reconstruction loss on validation sets
 - 15: end for
 - 16: Extract unified latent representations using the shared encoder.
 - 17: Concatenate latent features to form unified latent dataset
-

The masking strategy guarantees that dataset-specific encoder–decoder pathways are updated only when valid samples are present, while the shared encoder is continuously optimized using information from all active datasets. This interleaved training process allows the shared encoder to progressively integrate complementary knowledge from heterogeneous domains, promoting the learning of aligned, robust, and generalizable latent representations across datasets.

11. Latent Feature Extraction

After training completion, dataset-specific encoder models are extracted from the shared autoencoder. Each encoder maps its corresponding dataset into the shared latent space, producing compact latent feature vectors, as shown in Equation (14) [65,66]. These latent representations [74,75] capture aligned, noise-reduced, and discriminative characteristics of network traffic behavior across datasets.

$$Z^{(d)} = f_{encoder}(X^{(d)}) \quad (14)$$

where $Z^{(d)}$ is the latent representation of dataset d produced by the shared encoder $f_{encoder}$ and $X^{(d)}$ is the preprocessed input feature matrix of dataset d , containing cleaned, encoded, and normalized features.

The shared latent space provides a non-superficial representation of network threats by mapping heterogeneous features to universal security concepts. As demonstrated in Table 6, each latent dimension specializes in a distinct behavioral signature. For example, LD 9 consistently identifies Volumetric Saturation by aggregating dataset-specific stress indicators, such as SYN Flag Count in CSE-CIC-IDS2018 and OUT_PKTS in NF-BoT-IoT-v2. This cross-dataset consistency proves that the model learns the underlying physics of network congestion rather than superficial statistical correlations. Furthermore, the model effectively distinguishes between the characteristic patterns of DDoS and DoS attacks. While high-volume DDoS-UDP events are captured through the saturation metrics in LD 9 and LD 12, “low-and-slow” DoS attacks like Slowloris are isolated within LD 10 and LD 14, which focus on temporal inter-arrival anomalies (Flow IAT Min) and response payload characteristics. This clear semantic separation confirms that the latent space encodes the inherent mechanics of diverse attack categories, enabling robust generalization across PCAP, Netflow, and IoT traffic environments.

Table 6. Latent dimensions, top attacks, key features, and semantic interpretations.

Latent	Top 3 Associated Attacks	Representative Features (CSE-CIC, NF, IoT-23)	Semantic Interpretation
LD 1	DDoS-UDP, Theft, Recon	Fwd IAT Std, RETRAN_IN_PKTS, orig_bytes	Connection Initiation Dynamics
LD 2	DDoS-HTTP, Slowloris, Slow-Golden-Eye	Flow IAT Min, SRC_TO_DST_AVG_THR, conn_state	Temporal Flow Persistence
LD 3	Bot, DDoS-HOIC, Infiltration	Subflow Bwd Packets, CLIENT_FLAGS, orig_bytes	Traffic Burstiness & Variance
LD 4	DDoS-HOIC, Bot, SSH-Bruteforce	Bwd IAT Total, CLIENT_TCP_FLAGS, id_orig_h	Protocol Signaling Patterns

LD 5	DoS-Hulk, Theft, SQL Injection	Fwd Pkt Len Min, PKTS_128_TO_256, orig_bytes	Payload Sizing & Volume
LD 6	DDoS-UDP, Slowloris, Attack	Bwd Packets/s, NUM_PKTS_256_TO_512, history	Asymmetric Timing Fluctuations
LD 7	DoS, DDoS-UDP, Recon	Bwd IAT Mean, TCP_WIN_MAX_OUT, duration	Bidirectional Load Symmetry
LD 8	SSH-Bruteforce, DDoS-HOIC, Bot	Flow IAT Std, IN_PKTS, conn_state	Temporal Jitter & Flow State
LD 9	DDoS-UDP, DoS, Recon	SYN Flag Count, OUT_PKTS, id_resp_p	Volumetric Saturation & Stress
LD 10	Slowloris, Attack, SSH-Bruteforce	Flow IAT Min, L4_DST_PORT, id_resp_p	Temporal & Target Dynamics
LD 11	DoS-Hulk, Infiltration, SQL Injection	Bwd IAT Min, SERVER_TCP_FLAGS, history	Inter-Arrival Timing Consistency
LD 12	DDoS-UDP, Theft, Recon	Bwd IAT Mean, L4_DST_PORT, id_resp_p	Transmission Window Dynamics
LD 13	Brute Force-Web, SSH, Attack	Bwd IAT Std, L7_PROTO, orig_bytes	Request-Response Cycle Dynamics
LD 14	Slowloris, Slow-GoldenEye, Okiru	Fwd Pkts Len Tot, ICMP_IPV4_TYPE, resp_ip_bytes	Response Latency Profiling
LD 15	C&C, Brute Force-Web, C&C-PartOfA-HPS	Fwd Act Data Pkts, MIN_TTL, orig_bytes	Periodic Signaling (Beaconing)

The effectiveness of the shared autoencoder in learning meaningful latent features is reflected in the significant reduction in validation losses (MSE) for each dataset. The initial validation losses were 0.007737 for CSE-CIC-IDS2018, 0.000956 for NF-BoT-IoT-v2, and 0.004334 for IoT-23, which decreased to exceptionally low final losses of 0.000444, 0.000056, and 0.000127, respectively (Table 7). These results indicate that the autoencoder accurately reconstructs inputs while preserving dataset-specific characteristics and shared cross-domain patterns, producing robust, noise-reduced latent representations suitable for downstream classification tasks.

Table 7. Convergence of the shared autoencoder: initial vs. final validation loss across datasets.

Dataset	Initial Loss (Val)	Final Loss (Val)
CSE-CIC-IDS2018	0.007737	0.000444
NF-BoT-IoT-v2	0.000956	0.000056
IoT-23	0.004334	0.000127

12. Unified Latent Space Construction

The latent representations (15 latent spaces) obtained from all datasets are vertically concatenated to form a unified latent feature matrix, as shown in Equations (15) and (16) [57,62,76]. Corresponding labels from each dataset are merged accordingly. The theoretical rationale for this unification is to validate the Cross-Domain Alignment achieved during the autoencoder phase. Because the shared encoder has already mapped heterogeneous inputs into a compatible manifold, vertical concatenation does not result in feature mismatch. Instead, it creates a robust, multi-domain training set that allows the downstream classifier to learn attack signatures that are independent of the source network (IoT, Cloud, or Enterprise). This unified latent space enables the training of a single supervised classifier capable of performing generalized intrusion detection without requiring domain-specific fine-tuning.

$$Z_{Combined} = Z^{(CSE-CIC-IDS2018)} \parallel Z^{(NF-BoT-IoT-v2)} \parallel Z^{(IoT-23)} \quad (15)$$

$$y_{Combined} = y^{(CSE-CIC-IDS2018)} \parallel y^{(NF-BoT-IoT-v2)} \parallel y^{(IoT-23)} \quad (16)$$

where Z denotes the dataset-specific latent representations produced by encoders with shared weights, ensuring that all embeddings reside within a compatible common latent space, while y corresponds to the original labels of each dataset used for supervised classification.

13. Latent Feature Persistence

Finally, all latent feature matrices and corresponding labels are saved for subsequent classification experiments. The theoretical rationale for this persistence stage is to establish a “Decoupled Modular Framework.” By separating the feature extraction (manifold learning) from the classification stage, we ensure that the 15-dimensional behavioral signatures are stable and objective representations of network state. This modularity allows the learned representations to be reused efficiently across various classification architectures, such as DNN, CNN, or hybrid CNN–DNN models, without retraining the shared encoder. This demonstrates the scalability and generalizability of the proposed framework, as the same behavioral manifold can support multiple downstream security tasks, from binary detection to multi-class attack categorization.

14. Outlier Detection and Elimination Using LOF and Z-Score

The distribution of samples per class before and after the combined application of Z-score and LOF-based outlier removal, as presented in Table 8, demonstrates the impact of eliminating anomalous and noisy instances while preserving a robust training dataset. Several attack classes exhibit noticeable reductions in sample counts, indicating the effective removal of extreme or locally inconsistent observations. For instance, DDoS attacks–LOIC-HTTP decreased from 36,309 to 32,945 samples, DDOS attack–HOIC from 32,378 to 26,952, DoS attacks–Hulk from 30,570 to 27,905, and Bot traffic from 28,380 to 26,145 samples. Similar trends are observed across other attack categories, reflecting the ability of Z-score to filter global statistical outliers and LOF to detect local density-based anomalies. Overall, the joint Z-score and LOF-based data cleaning strategy enhances dataset quality by reducing noise and abnormal behavior while preserving essential information required for accurate, stable, and reliable intrusion detection.

Table 8. Distribution of samples per class before and after Z-score and LOF-based outlier removal.

Class	Before Z-Score & LOF	After Z-Score & LOF
Benign	108,863	5732
DDoS attacks-LOIC-HTTP	36,309	32,945
DDOS attack-HOIC	32,378	26,952
DoS attacks-Hulk	30,570	27,905
Bot	28,380	26,145
Infiltration	6600	4557
SSH-Bruteforce	20,755	16,059
DoS attacks-GoldenEye	16,918	13,774
DoS attacks-Slowloris	7623	5910
DDOS attack-LOIC-UDP	1680	1567
Brute Force-Web	471	305
Brute Force-XSS	202	139
SQL Injection	38	6
DoS attacks-SlowHTTPTest	47	32
FTP-BruteForce	45	32
Reconnaissance	61,420	32,806
DDoS	143,002	134,946

DoS	76,485	73,042
Theft	1739	1557
PartOfAHorizontalPortScan	69,104	2343
Okiru	14,930	14,117
C&C-HeartBeat	8132	7742
C&C	6711	6096
Attack	2122	1825
C&C-PartOfAHorizontalPortScan	236	103

15. Normalization of the Combined Dataset Latent Spaces

After generating and integrating the latent representations from the individual datasets, Min–Max normalization is applied only to the combined latent feature space before it is used as input to the learning framework. This step rescales all latent features to a unified range between 0 and 1 while preserving the original dimensionality of the latent vectors. Normalizing exclusively at the latent level is necessary because latent spaces derived from different datasets and encoders may exhibit heterogeneous value distributions and scales; without normalization, latent dimensions with larger magnitudes could disproportionately influence the learning process. Applying Min–Max scaling at this stage ensures scale consistency across latent features, improves numerical stability during training, and enables faster and more reliable convergence of downstream machine learning and deep learning models.

16. Splitting into Train and Test File

The dataset is split into training and testing sets using a flow-level unique-feature strategy. Each network flow is converted into a hashable feature vector, and the split is performed on the set of unique feature representations, resulting in zero overlap of duplicate or identical flows between the training and testing sets, such that the test set remains fully unseen during training. An 85%/15% train–test split is applied, effectively eliminating data leakage caused by duplicated flows. This splitting procedure is applied consistently across all classes, producing the per-class distributions reported in Table 9, where both frequent and rare attack categories are preserved to support robust learning and realistic evaluation under class imbalance. Although the split does not explicitly enforce temporal ordering, the strict zero-overlap constraint at the flow level guarantees statistical independence between training and testing data, providing a reliable assessment of the model's generalization performance across diverse network traffic behaviors.

Table 9. Train-test split of samples per class in the dataset.

Class	Train	Test
Benign	4888	844
DDoS attacks-LOIC-HTTP	27,916	5029
DDOS attack-HOIC	22,902	4050
DoS attacks-Hulk	23,698	4207
Bot	22,201	3944
Infiltration	3831	726
SSH-Bruteforce	13,671	2388
DoS attacks-GoldenEye	11,729	2045
DoS attacks-Slowloris	5003	907
DDOS attack-LOIC-UDP	1350	217
Brute Force-Web	251	54
Brute Force-XSS	116	23
SQL Injection	5	1

DoS attacks-SlowHTTPTest	28	4
FTP-BruteForce	28	4
Reconnaissance	27,959	4847
DDoS	114,662	20,284
DoS	62,182	10,860
Theft	1329	228
PartOfAHorizontalPortScan	1968	375
Okiru	11,985	2132
C&C-HeartBeat	6605	1137
C&C	5192	904
Attack	1558	267
C&C-PartOfAHorizontalPortScan	86	17

17. Class Imbalance Mitigation

Class imbalance in the training file was addressed using resampling techniques. ADASYN oversampled minority classes, while ENN removed noisy or borderline samples, improving data quality and supporting more robust model training.

(i) ADASYN

The effect of ADASYN on the binary classification dataset is illustrated in Table 10. The minority class, Normal, was oversampled from 4888 to 366,213 instances to match the majority Attack class, which remained at 366,255 samples. This resampling balances the dataset and helps the model learn effectively from both classes, reducing bias toward the originally dominant class.

Table 10. Distribution of samples per class before and after ADASYN on binary classification.

Class	Before ADASYN	After ADASYN
Normal	4888	366,213
Attack	366,255	366,255

The impact of ADASYN on the multi-class dataset is summarized in Table 11. While most majority classes remained unchanged, several minority classes were significantly oversampled to balance the dataset. For instance, SQL Injection and DoS attacks–Slow-HTTPTest increased dramatically from 5 and 28 samples to 114,661 and 114,661, respectively, and FTP–BruteForce was expanded from 28 to 114,664 samples. This resampling strategy ensures a more uniform class distribution, mitigating the bias toward originally dominant classes and enabling the model to learn effectively across all attack categories.

Table 11. Distribution of samples per class before and after ADASYN on multi-class classification.

Class	Before ADASYN	After ADASYN
Benign	4888	4888
DDoS attacks-LOIC-HTTP	27,916	27,916
DDOS attack-HOIC	22,902	22,902
DoS attacks-Hulk	23,698	23,698
Bot	22,201	22,201
Infiltration	3831	3831
SSH-Bruteforce	13,671	13,671
DoS attacks-GoldenEye	11,729	11,729
DoS attacks-Slowloris	5003	5003
DDOS attack-LOIC-UDP	1350	1350
Brute Force-Web	251	251

Brute Force-XSS	116	116
SQL Injection	5	114,661
DoS attacks-SlowHTTPTest	28	114,661
FTP-BruteForce	28	114,664
Reconnaissance	27,959	27,959
DDoS	114,662	114,662
DoS	62,182	62,182
Theft	1329	1329
PartOfAHorizontalPortScan	1968	1968
Okiru	11,985	11,985
C&C-HeartBeat	6605	6605
C&C	5192	5192
Attack	1558	1558
C&C-PartOfAHorizontalPortScan	86	86

(ii) ENN

The effect of ENN cleaning on the binary dataset is presented in Table 12. ENN slightly reduced the number of attack samples, decreasing them from 366,255 to 365,706, while the number of normal samples remained unchanged at 366,213. This minor reduction reflects the removal of noisy or borderline attack instances, improving the dataset quality and helping the model learn more robust decision boundaries between normal and attack traffic.

Table 12. Distribution of samples per class before and after ENN on binary classification.

Class	Before ENN	After ENN
Normal	366,213	366,213
Attack	366,255	365,706

The impact of ENN cleaning on the multi-class dataset is summarized in Table 13. ENN slightly reduced the number of samples across several classes to remove noisy or borderline instances, while leaving others unchanged. For example, benign samples decreased from 4888 to 4503, DDoS attacks–LOIC-HTTP from 27,916 to 27,882, and Bot traffic from 22,201 to 22,188. Certain minor classes, such as Brute Force-XSS and DDOS attack–LOIC-UDP, saw more noticeable reductions, reflecting the removal of inconsistent or low-density examples. Overall, ENN helped improve the dataset quality by eliminating potentially confusing or anomalous samples, supporting more robust training and enhancing the model’s ability to generalize across the 25 traffic classes.

Table 13. Distribution of samples per class before and after ENN on multi-class classification.

Class	Before ENN	After ENN
Benign	4888	4503
DDoS attacks-LOIC-HTTP	27,916	27,882
DDOS attack-HOIC	22,902	22,902
DoS attacks-Hulk	23,698	23,698
Bot	22,201	22,188
Infiltration	3831	3831
SSH-Bruteforce	13,671	13,669
DoS attacks-GoldenEye	11,729	11,729
DoS attacks-Slowloris	5003	5003
DDOS attack-LOIC-UDP	1350	1267

Brute Force-Web	251	227
Brute Force-XSS	116	66
SQL Injection	114,661	114,659
DoS attacks-SlowHTTPTest	114,661	15,755
FTP-BruteForce	114,664	15,461
Reconnaissance	27,959	27,832
DDoS	114,662	113,703
DoS	62,182	60,718
Theft	1329	1329
PartOfAHorizontalPortScan	1968	1477
Okiru	11,985	11,985
C&C-HeartBeat	6605	6605
C&C	5192	5186
Attack	1558	1553
C&C-PartOfAHorizontalPortScan	86	86

3.3. Proposed CNN–DNN Model

This section describes the CNN–DNN model configuration and hyperparameters. The model employs CNN layers to efficiently extract features from the input data and DNN layers to perform robust classification across diverse datasets.

3.3.1. Convolutional Neural Networks–Deep Neural Network (CNN–DNN)

The proposed hybrid CNN–DNN architecture is designed to efficiently extract high-level representations from network traffic data and perform accurate intrusion detection across both binary and multi-class tasks. CNN layers capture local spatial dependencies and patterns within the feature space, while the DNN layers perform hierarchical abstraction and classification. This combination allows the model to balance spatial sensitivity with deep decision learning, offering strong generalization across heterogeneous intrusion datasets.

3.3.2. Architectural Overview

The proposed model begins with an input layer defined by the number of features in the dataset. The input is processed through two convolutional blocks, each consisting of one-dimensional convolutional layers followed by batch normalization, ReLU activation, max pooling, and dropout for regularization. The extracted feature maps are then flattened and passed into two fully connected layers. Each dense layer is followed by batch normalization, ReLU activation, and dropout to improve generalization and reduce overfitting. For binary classification, the output layer consists of a single neuron with a sigmoid activation function, while for multi-class classification, the output layer applies a softmax activation function, where the number of output neurons corresponds to the number of target classes. This finalized architecture ensures that the proposed model is well-suited for both binary and multi-class classification tasks across diverse intrusion detection datasets.

In order to extract features, the CNN layers apply convolution kernels to the input data, slide over the feature map, and compute dot products to create new feature maps, as shown mathematically in Equation (17) [64].

$$Z_{i,j} = (X * K)_{i,j} = \sum_m \sum_n Z_{i+m,j+n} k_{m,n} \quad (17)$$

The input feature map is shown by X in this case, and the output feature map by Z . K is the convolution kernel, which is used to convert the input features into the output features during the convolution process.

Equation (18) [77] illustrates how the ReLU activation function mitigates vanishing gradients and encourages sparse activations to enhance training effectiveness and model performance by producing zero for negative inputs and retaining positive values.

$$\text{ReLU}(x) = \max(0, x) \quad (18)$$

Max pooling reduces spatial dimensions while maintaining important features by downsampling the input feature map by choosing the maximum value inside a pooling window, as Equation (19) [64] illustrates.

$$P_{i,j} = \max(X_{i:i+p, j:j+q}) \quad (19)$$

where P stands for the pooled output that was produced by the pooling operation, and p and q are the sizes of the pooling window that was used to combine the input features into the pooled output.

A proportion p of input units is randomly set to zero during training by the dropout layer in order to avoid overfitting. By ensuring that the model does not rely too strongly on any one input characteristic, as shown in Equation (20) [78], this strategy serves to increase the model's generalization.

$$\text{Dropout}(x) = \begin{cases} x & \text{with probability } 1 - p \\ 0 & \text{with probability } p \end{cases} \quad (20)$$

In order to generate the output, the feedforward process in a DNN entails transferring the input through several layers. This may be quantitatively represented for a layer l using Equation (21) [79].

$$a^{(l)} = f(W^{(l)} a^{(l-1)} + b^{(l)}) \quad (21)$$

The activation of the current layer l is shown in this case by $a^{(l)}$. Whereas $b^{(l)}$ is the bias vector for layer l , $W^{(l)}$ represents the weight matrix for this layer. Non-linearity is introduced into the model by applying the activation function f element-wise, which may involve functions like sigmoid or ReLU.

The output layer uses the sigmoid function for binary classification tasks, producing a probability score that shows how likely an instance is to belong to the positive class. Equation (22) [80] is the mathematical representation of this.

$$\sigma(Z) = \frac{1}{1 + e^{-Z}} \quad (22)$$

A probability distribution spanning several classes may be generated by the model thanks to the output layer's use of the softmax function for multi-class classification. Equation (23) [80] provides a mathematical representation for this.

$$\text{Softmax}(Z_i) = \frac{e^{Z_i}}{\sum_j e^{Z_j}} \quad (23)$$

Z is the output of the final dense layer in this case.

(i) Binary Classification

This versatile CNN–DNN model is specifically designed for binary classification, adjusting to various datasets by configuring its input layer to the number of features in each: 15 for the combined dataset, 67 for CSE-CIC-IDS2018, 41 for NF-BoT-IoT-v2, and 120 for IoT-23. The CNN section includes two convolutional blocks, both using 256 filters, with kernel sizes of 2 and 4, respectively. Each block is followed by batch normalization, ReLU activation, max pooling, and dropout (0.0000001) to enhance performance. The flattened output is then passed to the DNN component, which consists of two fully connected layers with 1024 and 768 neurons. These layers incorporate L2 regularization (0.0001), batch normalization, ReLU activation, and dropout (0.0000001) to prevent overfitting, as shown in Table 14. A single-neuron output layer with a sigmoid activation function concludes

the model, effectively distinguishing between Normal and Attack classes. This architecture allows for robust feature extraction and classification, ensuring strong generalization across different network intrusion detection datasets.

Table 14. Structure of the CNN–DNN model for binary classification.

Model Stage	Block	Layers	Layer Size	Activation
CNN	Input block	Input layer	Number of features	-
		1D CNN layer	256	-
	Hidden block 1	Batch normalization	-	ReLU
		1D Max pooling layer	2	-
		Dropout layer	0.0000001	-
		1D CNN layer	256	-
	Hidden block 2	Batch normalization	-	ReLU
		1D Max pooling layer	4	-
		Dropout layer	0.0000001	-
DNN	Hidden block 3	Dense layer	1024	-
		Batch normalization	-	ReLU
		Dropout layer	0.0000001	-
	Hidden block 4	Dense layer	768	-
		Batch normalization	-	ReLU
		Dropout layer	0.0000001	-
	Output block	Output layer	1 (Binary)	Sigmoid

(ii) Multi-class Classification

This adaptable hybrid CNN–DNN model is specifically designed for multi-class classification across various datasets. Its input layer is customized to the dimensionality of the specific dataset being used: 15 features for the combined dataset, 67 for CSE-CIC-IDS2018, 41 for NF-BoT-IoT-v2, and 120 for IoT-23. The model's CNN section includes two sequential convolutional blocks. The first block uses 256 filters with a kernel size of 2, while the second uses the same number of filters but with a larger kernel size of 4. Both blocks are followed by batch normalization, ReLU activation, max pooling, and dropout (0.0000001) to enhance performance, as shown in Table 15. The flattened output from the CNN is then directed to the DNN component, which features two fully connected layers containing 1024 and 768 neurons, respectively. These layers incorporate L2 regularization (0.0001), batch normalization, ReLU activation, and dropout (0.0000001) to prevent overfitting. The final softmax output layer adjusts its neuron count based on the number of classes in the dataset: 25 for the combined dataset, 15 for CSE-CIC-IDS2018, 5 for NF-BoT-IoT-v2, and 8 for IoT-23, ensuring effective classification.

Table 15. Structure of the CNN–DNN model for multi-class classification.

Model Stage	Block	Layers	Layer Size	Activation
CNN	Input block	Input layer	Number of features	-
		1D CNN layer	256	-
	Hidden block 1	Batch normalization	-	ReLU
		1D Max pooling layer	2	-
		Dropout layer	0.0000001	-
		1D CNN layer	256	-
	Hidden block 2	Batch normalization	-	ReLU
		1D Max pooling layer	4	-
		Dropout layer	0.0000001	-

DNN	Hidden block 3	Dense layer	1024	-
		Batch normalization	-	ReLU
		Dropout layer	0.0000001	-
	Hidden block 4	Dense layer	768	-
		Batch normalization	-	ReLU
		Dropout layer	0.0000001	-
	Output block	Output layer	Number of classes (Multi-class)	Softmax

(iii) Setting Up the CNN–DNN Models Hyperparameters

The Adam optimizer was used to set up the model for both binary and multi-class classification, with a batch size of 128, as shown in Table 16. If validation performance did not improve, the learning rate (ReduceLROnPlateau scheduler) was cut in half with a bottom restriction of 1×10^{-5} . At 0.001, it started. Accuracy was used as the assessment metric, binary cross-entropy for two-class issues, and categorical cross-entropy for multi-class tasks in order to ensure robust learning across datasets.

Table 16. The CNN–DNN Models Hyperparameters.

Parameter	Binary Classifier	Multi-Class Classifier
Batch size	128	128
Learning rate	Scheduled: Initial = 0.001, Factor = 0.5, Min = 1×10^{-5} (ReduceLROnPlateau)	Scheduled: Initial = 0.001, Factor = 0.5, Min = 1×10^{-5} (ReduceLROnPlateau)
Optimizer	Adam	Adam
Loss function	Binary_crossentropy	Categorical_crossentropy
Metric	Accuracy	Accuracy

4. Results and Experiments

This section provides an overview of the datasets and preprocessing steps, followed by the baseline models. It then presents the experimental results of the proposed CNN–DNN framework, including comparisons with existing methods on the CSE-CIC-IDS2018, NF-BoT-IoT-v2, IoT-23, and combined datasets for both binary and multi-class classification. Finally, a comprehensive ablation and sensitivity analysis is conducted, along with comparisons with recent models and evaluations of time, memory, and model size.

4.1. Overview of Dataset Properties and Preprocessing

The study utilizes four datasets (Three source datasets and one unified cross-domain dataset) with specific preprocessing for each. The CSE-CIC-IDS2018 dataset [48,49] includes 80 features across 15 classes (1 benign, 14 attacks) and was preprocessed by merging daily CSV files, removing duplicates, sampling, detecting outliers using LOF and Z-score, normalizing features, and splitting for training and evaluation. The NF-BoT-IoT-v2 dataset [50,51], with over 37 million flows heavily skewed toward malicious traffic (99.64%), underwent sampling, outlier detection using Z-score and LOF, MinMax scaling, and class imbalance handling using ADASYN, ENN, and class weighting. The IoT-23 dataset [11,26,52–55] comprises 23 scenarios (3 benign, 20 malicious) and was preprocessed similarly, with sampling, removal of incomplete and duplicate records, Z-score-based outlier elimination, MinMax scaling, splitting, and class imbalance mitigation using ADASYN, ENN, and class weighting to enhance model robustness. A unified dataset was then constructed by merging the three datasets through a shared autoencoder with dataset-specific projection layers and a 15-dimensional latent space. Dataset-specific decoders maintain reconstruction accuracy, while latent features are combined for cross-domain learning. The unified dataset includes 25 classes (1 benign, 24 attacks) and was further

processed for outliers, normalization, splitting, and class imbalance. All processed datasets were subsequently used for model training and evaluation.

4.2. Configuration and Hyperparameter Summary of the Comparable Models

This section outlines the setup and parameter settings for the standard models, such as CNN, autoencoder, and DNN [81–83].

- Convolutional Neural Networks (CNN)

The input features are processed by the model, and then three hidden convolutional layers with batch normalization, ReLU activation, max-pooling, and dropout are used [26]. The output layer for binary classification is a single neuron that can differentiate between attack and normal traffic using sigmoid activation. A softmax layer is employed for multi-class classification, yielding twenty-five classes for the combined dataset, fifteen for CSE-CIC-IDS2018, five for NF-BoT-IoT-v2, and eight for IoT-23, guaranteeing flexibility across various datasets and activities [64,84].

- Autoencoder

The model begins with unsupervised training of an autoencoder to generate compact representations of the input features that match the dataset's dimensions. Three dense layers (128, 64, and 32 neurons) with ReLU activation make up the encoder, while sigmoid activation is used in the decoder to symmetrically rebuild the input. Following training, the encoder is used as a feature extractor and linked to a classification layer, while the decoder is discarded [85–88]. For binary classification, the output layer consists of a single neuron with sigmoid activation to separate Normal and Attack traffic. A softmax layer is used for multi-class classification, yielding twenty-five outputs for the combined dataset, fifteen for CSE-CIC-IDS2018, five for NF-BoT-IoT-v2, and eight for IoT-23. This ensures flexibility and robustness across various datasets and tasks [22,26,28,30,44].

- Deep Neural Network (DNN)

The model receives a number of features at the input layer, which feeds into a dense layer of 1024 neurons using ReLU activation. The first hidden layer then applies dropout, a dense layer of 768 neurons with ReLU activation, and batch normalization. The second hidden layer uses both dropout and batch normalization [26,64]. Finally, the output layer classifies the data. For binary classification (e.g., Normal vs. Attack), a single neuron with a sigmoid activation is used. For multi-class classification, a softmax layer provides twenty-five outputs for the combined dataset, fifteen for CSE-CIC-IDS2018, five for NF-BoT-IoT-v2, and eight for IoT-23, making the model effective for various intrusion detection tasks.

Setting Up Hyperparameters for Models

The models were set up with the Adam optimizer with a batch size of 128 for both binary and multi-class classification. With a bottom constraint of 1×10^{-5} , learning rates (ReduceLROnPlateau scheduler) were halved if validation performance did not increase. They began at 0.001. To guarantee robust learning across datasets, binary cross-entropy was utilized for two-class tasks, categorical cross-entropy for multi-class tasks, and accuracy as the evaluation metric.

4.3. Establishment of the Experiment

Using Keras 3.10.0 and TensorFlow 2.19.0, we created the algorithmic structures in the Kaggle environment. Hardware running Windows 10 and equipped with an Nvidia GeForce RTX 1050 graphics card was used for the testing.

4.4. Evaluation Metrics

Model performance was evaluated using common metrics: accuracy, precision, recall, and F1-score. A combined dataset of CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23 was used to assess the model's generalization capability, while separate analyses were performed on each dataset to evaluate performance individually. The metrics are calculated based on true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), as shown in Equations (24)–(27) [89–95]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (24)$$

$$Precision = \frac{TP}{TP + FP} \quad (25)$$

$$Recall = \frac{TP}{TP + FN} \quad (26)$$

$$Fscore = \frac{2 \times precision \times recall}{precision + recall} \quad (27)$$

4.5. Results

The suggested CNN–DNN model's performance is assessed in this section by contrasting it with baseline models such as CNN, autoencoder, and DNN. The unified dataset was used for the tests in order to evaluate the model's overall efficacy. The CSE-CIC-IDS2018, NF-BoT-IoT-v2 and IoT-23 datasets were used to assess the models independently in order to confirm their capacity for generalization. The results show that the proposed approach is both robust and adaptable across diverse network traffic scenarios.

(i) Binary Classification

A comprehensive investigation of binary classification performance across the CSE-CIC-IDS2018, NF-BoT-IoT-v2, IoT-23, and combined datasets demonstrates the pronounced superiority of the proposed CNN–DNN framework, as detailed in Table 17. Within the CSE-CIC-IDS2018 dataset, the hybrid architecture achieves optimal outcomes, with 99.73% accuracy, 99.74% precision, 99.73% recall, and 99.73% F-score, slightly surpassing the standalone CNN and DNN models, both with 99.71% accuracy, and the autoencoder at 99.69%. In the NF-BoT-IoT-v2 dataset, the CNN–DNN sustains its dominant position with all evaluation metrics attaining 99.98%, outperforming the DNN at 99.96%, the CNN at 99.97%, and the autoencoder at 99.95%. The IoT-23 dataset further exemplifies the hybrid model's efficacy, with scores reaching 99.99%, exceeding the CNN and DNN at 99.97% and the autoencoder at 99.96%. When assessed on the combined dataset using a shared autoencoder, the proposed CNN–DNN model consistently demonstrates superior performance, achieving 99.76% accuracy, 99.80% precision, 99.76% recall, and 99.77% F-score, while individual models exhibit slightly lower effectiveness, with metrics ranging from 99.64% to 99.78%. Collectively, these findings highlight the CNN–DNN framework's robustness, versatility, and reliable generalization, affirming its capacity to deliver consistently exceptional binary classification performance across diverse intrusion detection scenarios.

Table 17. Comparison of performance metrics for binary classification models.

Dataset	Model	Accuracy	Precision	Recall	F-Score
CSE-CIC-IDS2018	CNN	99.71%	99.72%	99.71%	99.71%
	Autoencoder	99.69%	99.70%	99.69%	99.69%
	DNN	99.71%	99.72%	99.71%	99.71%

NF-BoT-IoT-v2	CNN–DNN (Proposed)	99.73%	99.74%	99.73%	99.73%
	CNN	99.97%	99.97%	99.97%	99.97%
	Autoencoder	99.95%	99.95%	99.95%	99.95%
	DNN	99.96%	99.96%	99.96%	99.96%
IoT-23	CNN–DNN (Proposed)	99.98%	99.98%	99.98%	99.98%
	CNN	99.97%	99.97%	99.97%	99.97%
	Autoencoder	99.96%	99.96%	99.96%	99.96%
	DNN	99.97%	99.97%	99.97%	99.97%
Combined Dataset (using shared autoencoder)	CNN–DNN (Proposed)	99.99%	99.99%	99.99%	99.99%
	CNN	99.74%	99.78%	99.74%	99.75%
	Autoencoder	99.72%	99.77%	99.72%	99.73%
	DNN	99.64%	99.71%	99.64%	99.66%
	CNN–DNN (Proposed)	99.76%	99.80%	99.76%	99.77%

(ii) Multi-Class Classification

An extensive assessment of multi-class classification approaches across the CSE-CIC-IDS2018, NF-BoT-IoT-v2, IoT-23, and combined datasets demonstrates the pronounced efficacy of the proposed CNN–DNN framework, as illustrated in Table 18. On the CSE-CIC-IDS2018 dataset, the hybrid architecture achieves superior performance, attaining 99.61% accuracy, 99.63% precision, 99.61% recall, and 99.60% F1-score, thereby outperforming the individual CNN, autoencoder, and DNN models, whose metrics range from 99.52% to 99.59%. For the NF-BoT-IoT-v2 dataset, the CNN–DNN maintains its leading position, achieving 98.28% accuracy, 98.29% precision, 98.28% recall, and 98.28% F1-score, slightly surpassing the CNN and DNN models, whose metrics range from 98.19% to 98.26%, while the autoencoder records slightly lower values, ranging from 98.21% to 98.25%. On the IoT-23 dataset, the hybrid model again demonstrates superior capability, achieving scores from 99.94% to 99.96%, outperforming both the CNN and DNN models, which range from 99.75% to 99.94%, as well as the autoencoder, which ranges from 99.54% to 99.58%. When evaluated on the combined dataset using the shared autoencoder, the proposed CNN–DNN model demonstrates the highest evaluation metrics, achieving 99.54% accuracy, 99.55% precision, 99.54% recall, and 99.54% F-score, while the standalone CNN and DNN models achieve slightly lower accuracies of 99.48% and 99.49%, and the autoencoder attains 99.46% accuracy. These results collectively highlight the CNN–DNN framework's robustness, versatility, and strong generalization ability across heterogeneous intrusion detection environments.

Table 18. Comparison of performance metrics for multi-class classification models.

Dataset	Model	Accuracy	Precision	Recall	F-Score
CSE-CIC-IDS2018	CNN	99.58%	99.59%	99.58%	99.57%
	Autoencoder	99.54%	99.54%	99.54%	99.52%
	DNN	99.57%	99.56%	99.57%	99.55%
	CNN–DNN (Proposed)	99.61%	99.63%	99.61%	99.60%
NF-BoT-IoT-v2	CNN	98.23%	98.26%	98.23%	98.23%
	Autoencoder	98.21%	98.25%	98.21%	98.21%
	DNN	98.19%	98.24%	98.19%	98.19%
	CNN–DNN (Proposed)	98.28%	98.29%	98.28%	98.28%
IoT-23	CNN	99.91%	99.94%	99.91%	99.92%
	Autoencoder	99.54%	99.58%	99.54%	99.55%
	DNN	99.75%	99.80%	99.75%	99.76%
	CNN–DNN (Proposed)	99.94%	99.96%	99.94%	99.95%

Combined Dataset (Using Shared Autoencoder)	CNN	99.48%	99.50%	99.48%	99.47%
	Autoencoder	99.46%	99.55%	99.46%	99.50%
	DNN	99.49%	99.49%	99.49%	99.49%
	CNN–DNN (Proposed)	99.54%	99.55%	99.54%	99.54%

4.6. Comprehensive Ablation and Sensitivity Analysis of the Proposed Framework

This section analyzes the proposed framework through an ablation study, the impact of resampling strategies, latent space dimensionality analysis, and evaluation under different random seed splits to assess performance contributions and robustness.

1. Proposed CNN–DNN Ablation Study

The ablation study of the proposed CNN–DNN model, summarized in Table 19, compares its performance with standalone CNN and DNN across four datasets: CSE-CIC-IDS2018, NF-BoT-IoT-v2, IoT-23, and the combined dataset. For binary classification, the CNN–DNN model achieves accuracies of 99.73%, 99.98%, 99.99%, and 99.76% on the respective datasets, while the CNN and DNN models range from 99.41% to 99.76%. In multi-class classification, the CNN–DNN model attains accuracies of 99.61%, 98.28%, 99.91%, and 99.54%, surpassing the CNN and DNN, which range from 89.38% to 99.44%. These results demonstrate the CNN–DNN model's consistent robustness and superior performance across both binary and multi-class intrusion detection tasks.

Table 19. Ablation study of the CNN–DNN model compared with standalone CNN and DNN.

Classification Type	Model	CSE-CIC-IDS2018	NF-BoT-IoT-v2	IoT-23	Combined
Binary	CNN	99.52%	99.76%	99.62%	99.54%
	DNN	99.41%	99.74%	94.85%	99.43%
	CNN–DNN(Proposed)	99.73%	99.98%	99.99%	99.76%
Multi-Class	CNN	99.41%	97.87%	93.61%	99.35%
	DNN	99.44%	97.91%	89.38%	99.29%
	CNN–DNN(Proposed)	99.61%	98.28%	99.91%	99.54%

2. Impact of Resampling Strategies

The impact of different resampling strategies on the CNN–DNN model is summarized in Table 20, showing their effect across all evaluated datasets. For the CSE-CIC-IDS2018 dataset, binary classification without resampling reaches 99.73%, while ENN records 99.66%, and the combined value of ADASYN and ENN reaches 99.54%. In NF-BoT-IoT-v2, the model benefits more from resampling: binary classification improves from 99.94% without resampling to 99.98% using ADASYN and ENN, and multi-class accuracy increases from 98.18% to 98.28%, highlighting the effectiveness of addressing class imbalance. For IoT-23, the combined strategy similarly provides the best performance for both binary 99.99% and multi-class tasks 99.94%, outperforming individual ADASYN or ENN approaches. On the combined dataset, which includes heterogeneous traffic from all sources, ADASYN and ENN consistently deliver the highest accuracy for binary 99.76% and multi-class 99.54% classification, while other strategies range from 99.71% to 99.73% for binary and 99.45% to 99.49% for multi-class tasks. These results indicate that although the CNN–DNN model performs robustly without resampling, applying resampling techniques can further enhance performance, particularly in addressing class imbalance across heterogeneous datasets.

Table 20. Effect of resampling strategies on binary and multi-class Tasks.

Classification Type	Resampling Strategy	CSE-CIC-IDS2018	NF-BoT-IoT-v2	IoT-23	Combined
Binary	No Resampling	99.73%	99.94%	99.91%	99.71%
	ADASYN	99.48%	99.97%	99.94%	99.73%
	ENN	99.66%	99.96%	99.95%	99.72%
	ADASYN + ENN	99.54%	99.98%	99.99%	99.76%
Multi-Class	No Resampling	99.61%	98.18%	99.89%	99.45%
	ADASYN	99.57%	98.20%	99.90%	99.49%
	ENN	99.56%	98.21%	99.86%	99.47%
	ADASYN + ENN	99.55%	98.28%	99.94%	99.54%

3. Latent Space Dimensionality Analysis

Evaluation of the shared autoencoder with different latent space dimensionalities shows that, for the binary task, accuracy is 99.58% with 10 latent dimensions, increases to 99.76% at 15 dimensions, and slightly decreases to 99.68% with 20 dimensions. For the multi-class task, accuracy is 99.43% for 10 dimensions, reaches 99.54% at 15 dimensions, and is 99.48% with 20 dimensions, as shown in Table 21. These results indicate that 15 latent dimensions provide the most informative and discriminative representation for both binary and multi-class classification tasks. Therefore, the proposed shared autoencoder-based framework with CNN–DNN classifier achieves optimal performance when the latent space is carefully tuned, confirming the robustness and effectiveness of the learned representations.

Table 21. Effect of latent space dimensionality on binary and multi-class Tasks.

Classification Type	Latent Space	Accuracy
Binary	10	99.58%
	15	99.76%
	20	99.68%
Multi-Class	10	99.43%
	15	99.54%
	20	99.48%

4. Analysis Under Different Random Seed Splits

Experiments were conducted using multiple random seeds affecting the dataset splitting process. Table 22 reports the classification accuracy for both binary and multi-class tasks under different random seeds. Across the four random seeds, the mean accuracy for binary classification is $99.76\% \pm 0.01\%$, and for multi-class classification is $99.54\% \pm 0.01\%$, indicating extremely small variation across runs. These results demonstrate that the proposed shared autoencoder-based framework with CNN–DNN classifier is robust and largely insensitive to random seed selection or data partitioning, thereby confirming the reliability, stability, and reproducibility of the reported performance.

Table 22. Effect of random seed splits for binary and multi-class classification tasks.

Classification Type	Random Seed (Splitting)	Accuracy
Binary	7	99.75%
	21	99.76%
	42	99.76%
	99	99.76%
Multi-Class	7	99.54%
	21	99.53%

	42	99.54%
	99	99.54%

4.7. Comparison with Recent Models

The performance comparison, summarized in Table 23, evaluates the proposed CNN–DNN model against recent approaches, including Transformer–CNN, CNN–MLP, and Transformer–DNN, for both binary and multi-class classification tasks. For binary classification, the CNN–DNN consistently achieves the highest accuracy across all datasets: 99.73% on CSE-CIC-IDS2018, 99.98% on NF-BoT-IoT-v2, 99.99% on IoT-23, and 99.76% on the combined dataset, surpassing other approaches whose accuracies range from 99.50% to 99.86%. In multi-class classification, the proposed model similarly demonstrates superior performance, attaining accuracies of 99.61%, 98.28%, 99.94%, and 99.54% on the respective datasets, exceeding other approaches that range from 95.17% to 99.80%. These results underscore the robustness, versatility, and enhanced generalization capability of the CNN–DNN framework in diverse intrusion detection scenarios.

Table 23. Comparison of the proposed CNN–DNN model with recent models.

Classification Type	Model	CSE-CIC-IDS2018	NF-BoT-IoT-v2	IoT-23	Combined
Binary	Transformer–CNN [30]	99.68%	99.83%	99.78%	99.66%
	CNN–MLP [26]	99.64%	99.79%	99.83%	99.68%
	Transformer–DNN [28]	99.66%	99.86%	99.50%	99.65%
	CNN–DNN (Proposed)	99.73%	99.98%	99.99%	99.76%
Multi-Class	Transformer–CNN [30]	99.43%	98.06%	99.80%	99.40%
	CNN–MLP [26]	99.52%	98.04%	99.79%	99.42%
	Transformer–DNN [28]	99.49%	98.02%	95.17%	99.39%
	CNN–DNN (Proposed)	99.61%	98.28%	99.94%	99.54%

4.8. Time, Memory, and Model Size

Model efficiency is crucial for real-time intrusion detection. This section evaluates the proposed CNN–DNN framework alongside baseline models in terms of inference time, training time, memory usage, and model size, demonstrating its suitability for time-sensitive and resource-constrained environments.

1. Inference time

Efficient inference is critical for real-time intrusion detection systems. The proposed CNN–DNN framework demonstrates extremely fast per-sample inference times of approximately 7.94×10^{-6} seconds (s) for binary classification and 7.96×10^{-6} s for multi-class classification, as shown in Table 24. These values fall within a similar range as the baseline models, roughly 6.0×10^{-6} to 8.0×10^{-6} s per sample, while providing superior classification performance. The results indicate that the model is highly efficient and well-suited for real-time deployment, delivering both high accuracy and practical usability in time-sensitive intrusion detection environments.

Table 24. Inference time.

Classification Type	Model	Inference Time (Seconds)
Binary	CNN	7.87902×10^{-6}
	Autoencoder	6.03750×10^{-6}
	DNN	6.29230×10^{-6}
	CNN–DNN(Proposed)	7.94129×10^{-6}
Multi-Class	CNN	7.99174×10^{-6}
	Autoencoder	7.54983×10^{-6}

DNN	7.84450×10^{-6}
CNN–DNN(Proposed)	7.96421×10^{-6}

2. Training time

Efficient training is essential for practical deployment of intrusion detection systems. The proposed CNN–DNN framework requires per-sample training times of approximately 9.33×10^{-6} s for binary classification and 9.45×10^{-6} s for multi-class classification, as shown in Table 25. These values are within a similar range as the baseline models, roughly 3.4×10^{-7} to 9.4×10^{-6} s per sample, demonstrating that the CNN–DNN framework can be trained efficiently while maintaining high performance. The results indicate that the model is well-suited for real-time or large-scale intrusion detection scenarios, providing both rapid training and practical deployment capability in operational environment.

Table 25. Training time.

Classification Type	Model	Training Time (Seconds)
Binary	CNN	9.152675×10^{-6}
	Autoencoder	3.364086×10^{-7}
	DNN	1.139832×10^{-6}
	CNN–DNN(Proposed)	9.329939×10^{-6}
Multi-Class	CNN	9.407687×10^{-6}
	Autoencoder	4.069567×10^{-7}
	DNN	1.233268×10^{-6}
	CNN–DNN(Proposed)	9.453416×10^{-6}

3. Memory consumption

Efficient memory usage is important for deploying intrusion detection systems on resource-constrained environments. The proposed CNN–DNN framework consumes approximately 0.134–0.136 megabytes (MB) per sample during inference and 0.267–0.270 MB per sample during training for binary and multi-class classification tasks, as shown in Table 26. These values fall within a reasonable range compared to the baseline models, demonstrating that the framework maintains efficient memory usage while providing high performance. The results indicate that the CNN–DNN model is suitable for real-time deployment and large-scale datasets, offering a practical balance between accuracy, speed, and resource requirements.

Table 26. Memory consumption.

Classification Type	Model	Memory Consumption (Inference) (MB)	Memory Consumption (Training) (MB)
Binary	CNN	0.0376	0.0750
	Autoencoder	0.0009	0.0017
	DNN	0.0074	0.0144
	CNN–DNN(Proposed)	0.1342	0.2670
Multi-Class	CNN	0.0390	0.0777
	Autoencoder	0.0009	0.0017
	DNN	0.0075	0.0146
	CNN–DNN(Proposed)	0.1356	0.2697

4. Model Size

Model size is a key factor for deployment, especially in environments with limited storage or memory capacity. The proposed CNN–DNN framework has a size of approximately 45.2 MB for binary classification and 45.4 MB for multi-class classification, as shown in Table 27. This size reflects the comprehensive and integrated design of the CNN–DNN, which combines feature extraction and classification in a single end-to-end framework. The model size remains practical for deployment and supports high-performance real-time intrusion detection, offering a balance between accuracy, efficiency, and resource requirements.

Table 27. Model Size.

Classification Type	Model	Size (MB)
Binary	CNN	6.11
	Autoencoder	0.17
	DNN	9.30
	CNN–DNN(Proposed)	45.20
Multi-Class	CNN	6.25
	Autoencoder	0.18
	DNN	9.51
	CNN–DNN(Proposed)	45.41

5. Discussion

A thorough evaluation of the proposed CNN–DNN model's performance is presented in this section. The discussion incorporates an examination of the confusion matrices derived from the Combined Dataset, CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23 datasets. Furthermore, a performance assessment is conducted for each individual class on every dataset, thereby illustrating the model's proficiency with varied traffic categories.

5.1. Binary Classification

The confusion matrices provide clear evidence of the proposed model's outstanding effectiveness in binary intrusion detection across all considered datasets, as reflected by the extremely low number of misclassified samples. In the CSE-CIC-IDS2018 dataset, the model correctly recognizes 887 normal instances and 36,404 attack instances, while generating only 31 false positives and 71 false negatives, as illustrated in Figure 2a. This outcome underscores the model's strong capability to distinguish between normal and attack traffic with minimal degradation in performance. For the NF-BoT-IoT-v2 dataset, shown in Figure 2b, the classifier correctly identifies 1744 normal samples and 11,416 attack samples, with only 1 false positive and 2 false negatives, demonstrating near-perfect discrimination between benign and malicious behaviors. A similarly remarkable trend is observed for the IoT-23 dataset in Figure 2c, where 10,654 normal flows and 23,102 attack flows are correctly classified, with no false positives and only 2 false negatives, highlighting the model's exceptional reliability in attack recognition. Finally, the results obtained on the combined dataset, presented in Figure 2d, indicate that the model successfully classifies 844 normal and 64,491 attack samples, while incurring 0 false positives and 159 false negatives, demonstrating its strong capability to accurately distinguish between benign and malicious traffic. These findings confirm that the proposed approach maintains high accuracy, robustness, and generalization capability even when operating on aggregated and heterogeneous data sources.

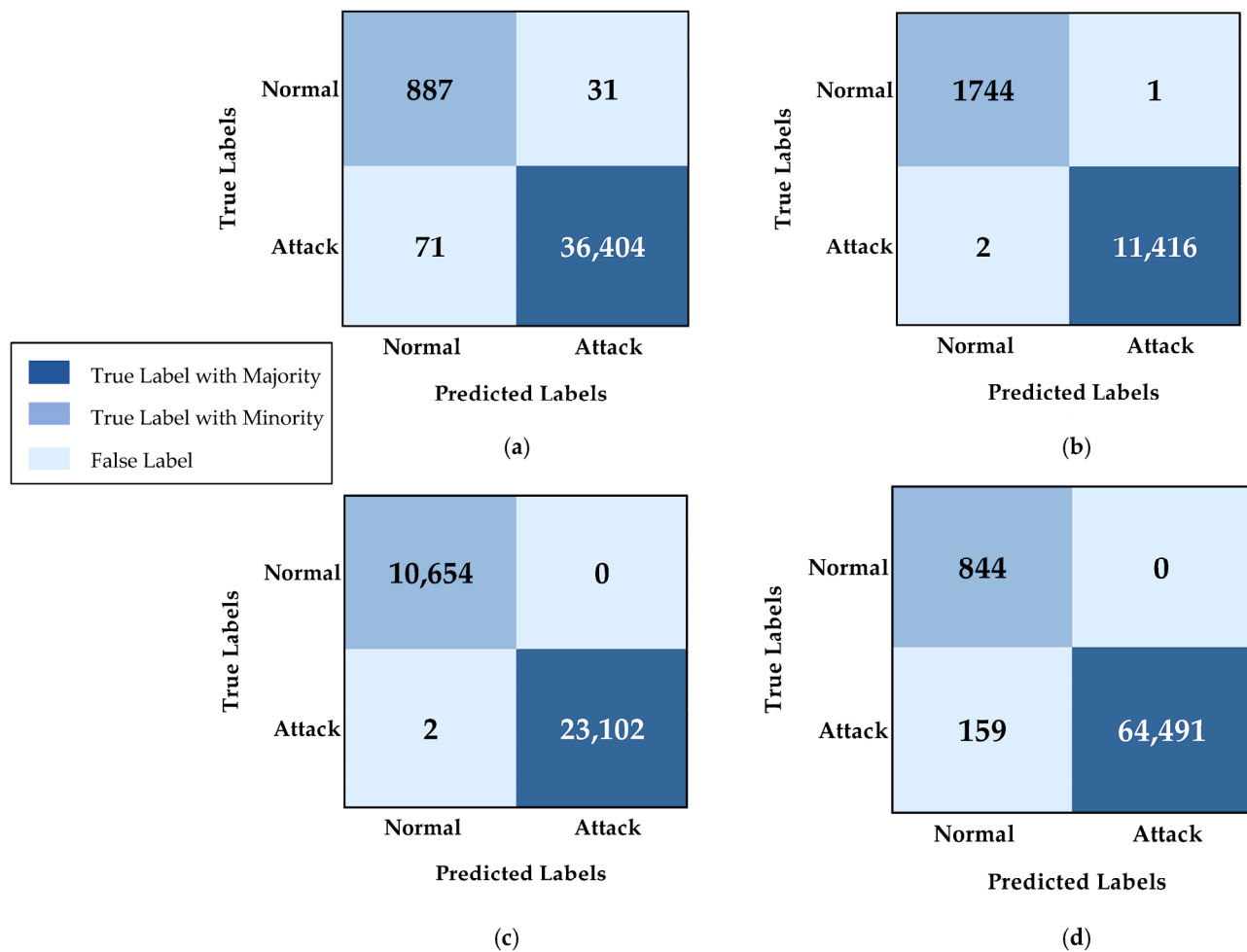


Figure 2. Confusion matrices for binary classification using the CNN-DNN model on (a) CSE-CIC-IDS2018, (b) NF-BoT-IoT-v2, (c) IoT-23, and (d) combined dataset (using shared autoencoder).

The comparison of binary classification accuracies across all evaluated datasets, illustrated in Figure 3, demonstrates the effectiveness of the proposed CNN-DNN framework. On the CSE-CIC-IDS2018 dataset, the hybrid model achieves the highest accuracy of 99.73%, slightly surpassing the CNN and DNN models at 99.71% and the autoencoder at 99.69%. A similar trend is observed for NF-BoT-IoT-v2, where the proposed CNN-DNN reaches 99.98%, outperforming CNN at 99.97%, autoencoder at 99.95%, and DNN at 99.96%. For IoT-23, the CNN-DNN attains 99.99%, exceeding CNN and DNN at 99.97% and autoencoder at 99.96%. Evaluated on the combined dataset generated using the shared autoencoder, the proposed model maintains superior performance with 99.76% accuracy, compared to CNN at 99.74%, autoencoder at 99.72%, and DNN at 99.64%. These results collectively highlight the robustness, stability, and strong generalization capability of the CNN-DNN architecture across both individual and aggregated heterogeneous intrusion detection datasets.

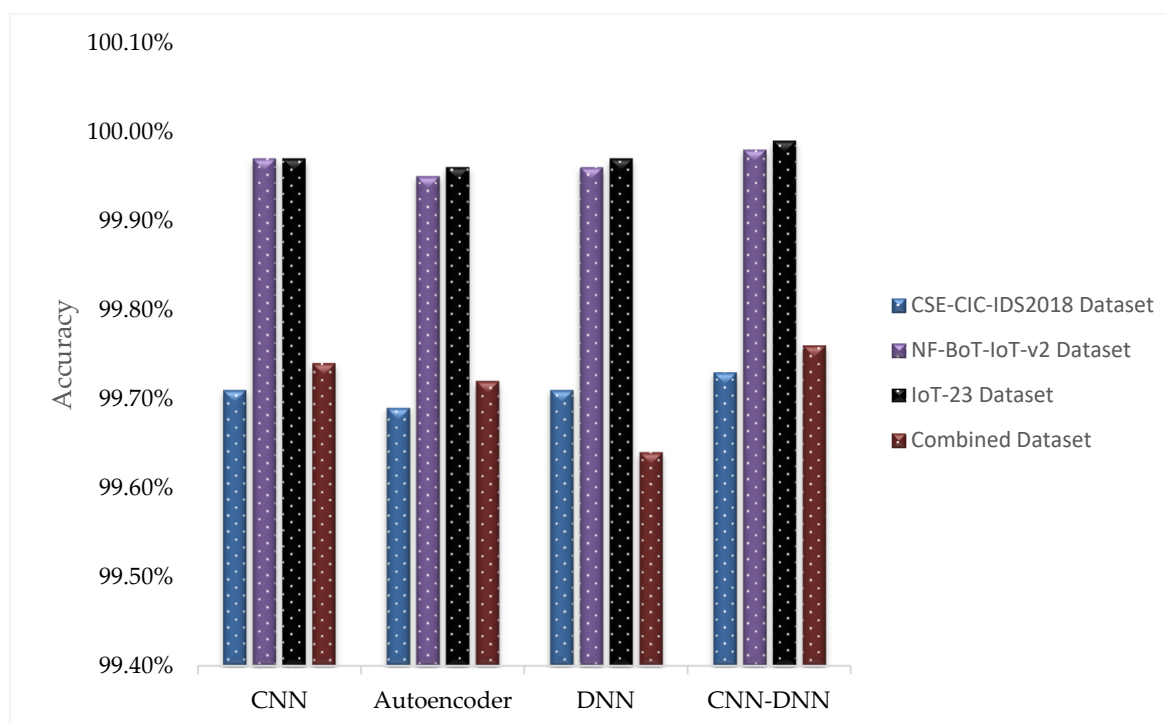


Figure 3. Proposed CNN–DNN versus binary classifiers.

The CNN–DNN framework demonstrates outstanding efficacy in binary classification tasks across a variety of datasets, as detailed in Table 28. Specifically, on the CSE-CIC-IDS2018 dataset, the model correctly identifies normal network traffic with an accuracy of 96.62%, while achieving 99.81% accuracy in detecting attack traffic, both accompanied by strong precision, recall, and F1-scores. Performance on the NF-BoT-IoT-v2 dataset approaches perfection, with the normal and attack categories accurately recognized at rates of 99.94% and 99.98%, respectively. Likewise, on the IoT-23 dataset, the model demonstrates almost flawless performance, accurately identifying normal traffic at 100% and attack traffic at 99.99%, with precision, recall, and F1-scores consistently remaining at very high levels. When assessed on the aggregated dataset using the shared autoencoder, the CNN–DNN model continues to exhibit superior performance, attaining 100% accuracy for benign traffic and 99.75% for malicious activity, along with uniformly high values of precision, recall, and F1-score. These findings highlight the model's robustness, its ability to generalize across heterogeneous network and IoT environments, and its potential as a reliable tool for comprehensive intrusion detection.

Table 28. Binary classification performance metrics for each class using the CNN–DNN model.

Dataset	Class	Accuracy	Precision	Recall	F-Score
CSE-CIC-IDS2018	Normal	96.62%	92.59%	96.62%	94.56%
	Attack	99.81%	99.91%	99.81%	99.86%
NF-BoT-IoT-v2	Normal	99.94%	99.89%	99.94%	99.91%
	Attack	99.98%	99.99%	99.98%	99.99%
IoT-23	Normal	100%	99.98%	100%	99.99%
	Attack	99.99%	100%	99.99%	100%
Combined Dataset (Using Shared Autoencoder)	Normal	100%	84.15%	100%	91.39%
	Attack	99.75%	100%	99.75%	99.88%

5.2. Multi-Class Classification

The multi-class confusion matrix for the CSE-CIC-IDS2018 dataset, evaluated using the proposed model, demonstrates exceptional precision and consistent discrimination across all traffic categories, as shown in Figure 4. Benign traffic is accurately identified in 892 instances, with only 26 misclassifications, while DDoS attacks-LOIC-HTTP and DDOS attack-HOIC are almost perfectly detected, with 7359 and 6457 correct classifications, respectively. DoS attacks-Hulk and DoS attacks-GoldenEye are effectively recognized, with 6125 and 3352 instances correctly classified. Additional attack types, including Bot, Infiltration, SSH-Bruteforce, and DoS attacks-Slowloris, show strong detection performance, with 5703, 1236, 4072, and 1588 correctly identified samples, respectively. Other classes, such as DDOS attack-LOIC-UDP, Brute Force-Web, Brute Force-XSS, SQL Injection, DoS attacks-SlowHTTPTest, and FTP-BruteForce, achieve correct classification counts of 350, 98, 11, 2, 1, and 2, respectively. The confusion matrix thus underscores the proposed model's robustness, reliability, and capability to accurately distinguish benign traffic from a wide spectrum of sophisticated cyberattacks, highlighting its strong effectiveness in multi-class intrusion detection on the CSE-CIC-IDS2018 dataset.

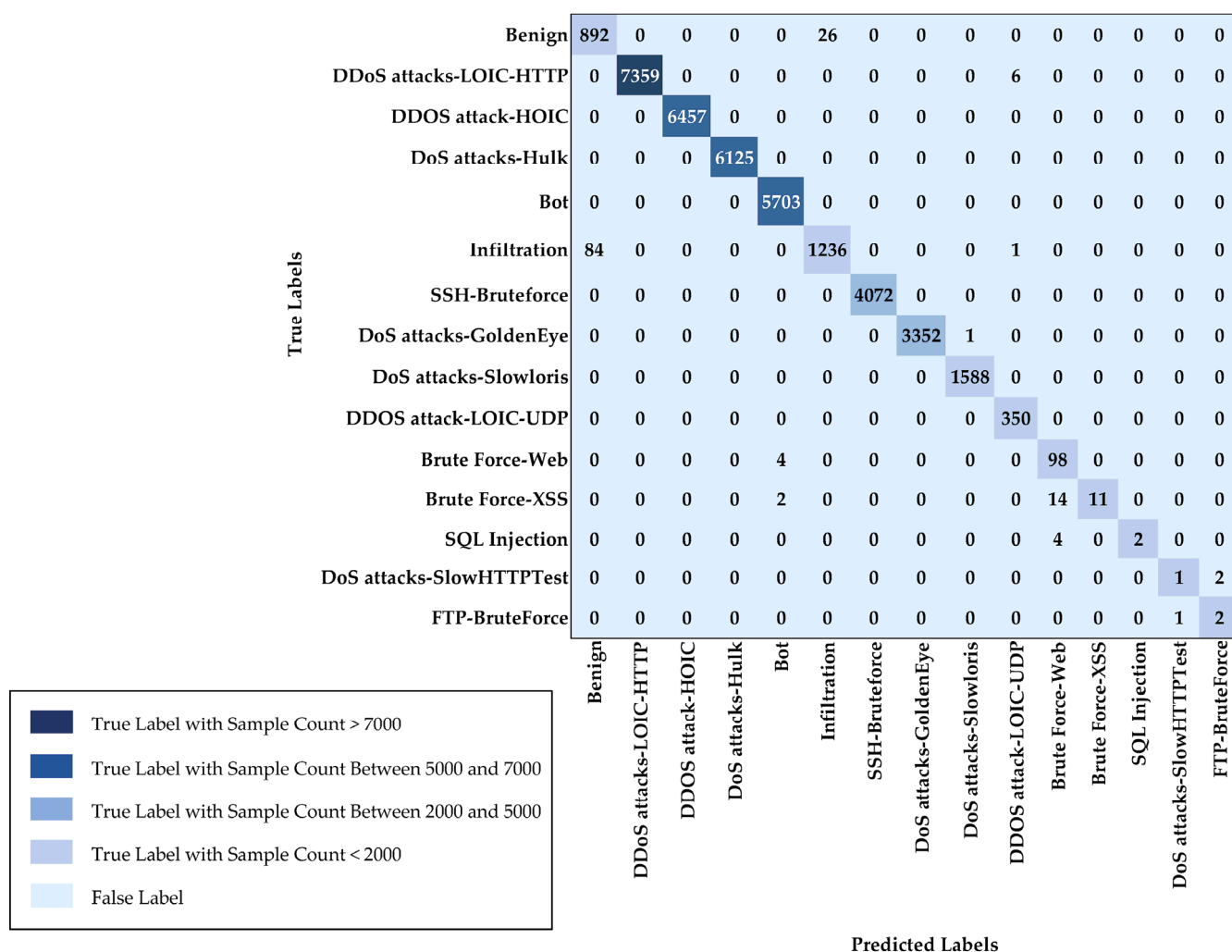


Figure 4. Confusion matrix for multi-class classification using the CNN-DNN model on CSE-CIC-IDS2018 dataset.

The proposed model is highly effective on the NF-BoT-IoT-v2 dataset at identifying benign traffic and theft, achieving 1744 and 97 correctly classified instances, respectively, as depicted in Figure 5. Strong performance is also evident in detecting high-volume attacks, with 4391 DDoS instances and 3814 DoS instances correctly identified. Reconnaissance traffic is largely detected, with 2890 correct classifications and a few misclassifications, including 123 instances as DoS, one as benign, and one as Theft. Overall, the model demonstrates robust and reliable performance in accurately categorizing a wide range of network activities, with only minimal errors across these areas. On the IoT-23, the model exhibits high classification accuracy across most categories, with large numbers representing correct predictions along the diagonal. For example, it correctly identified 10,654 instances of benign traffic and 8008 instances of DDoS traffic, as shown in Figure 6. Minor misclassifications are observed for certain classes, notably with 16 instances of PartOfAHorizontalPortScan being incorrectly predicted as a C&C-PartOfAHorizontalPortScan. The classifier also mislabeled a single instance of C&C traffic as benign and another as PartOfAHorizontalPortScan. The performance on smaller classes is also strong; for example, C&C-PartOfAHorizontalPortScan was correctly classified 27 times without any confusion with the larger categories. These findings demonstrate the model's capability to accurately recognize both majority and minority classes, underscoring its robustness in handling class imbalance. The limited number of misclassifications observed reflects the model's strong generalization ability, ensuring reliable performance across diverse traffic categories in multi-class classification tasks.

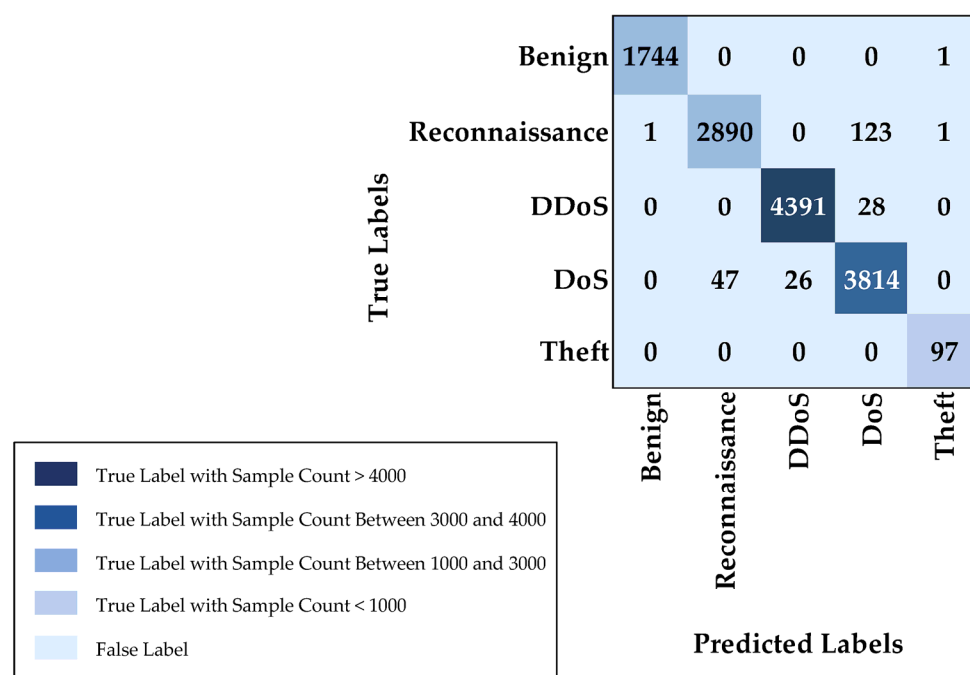


Figure 5. Confusion matrix for multi-class classification using the CNN-DNN model on NF-BoT-IoT-v2 dataset.

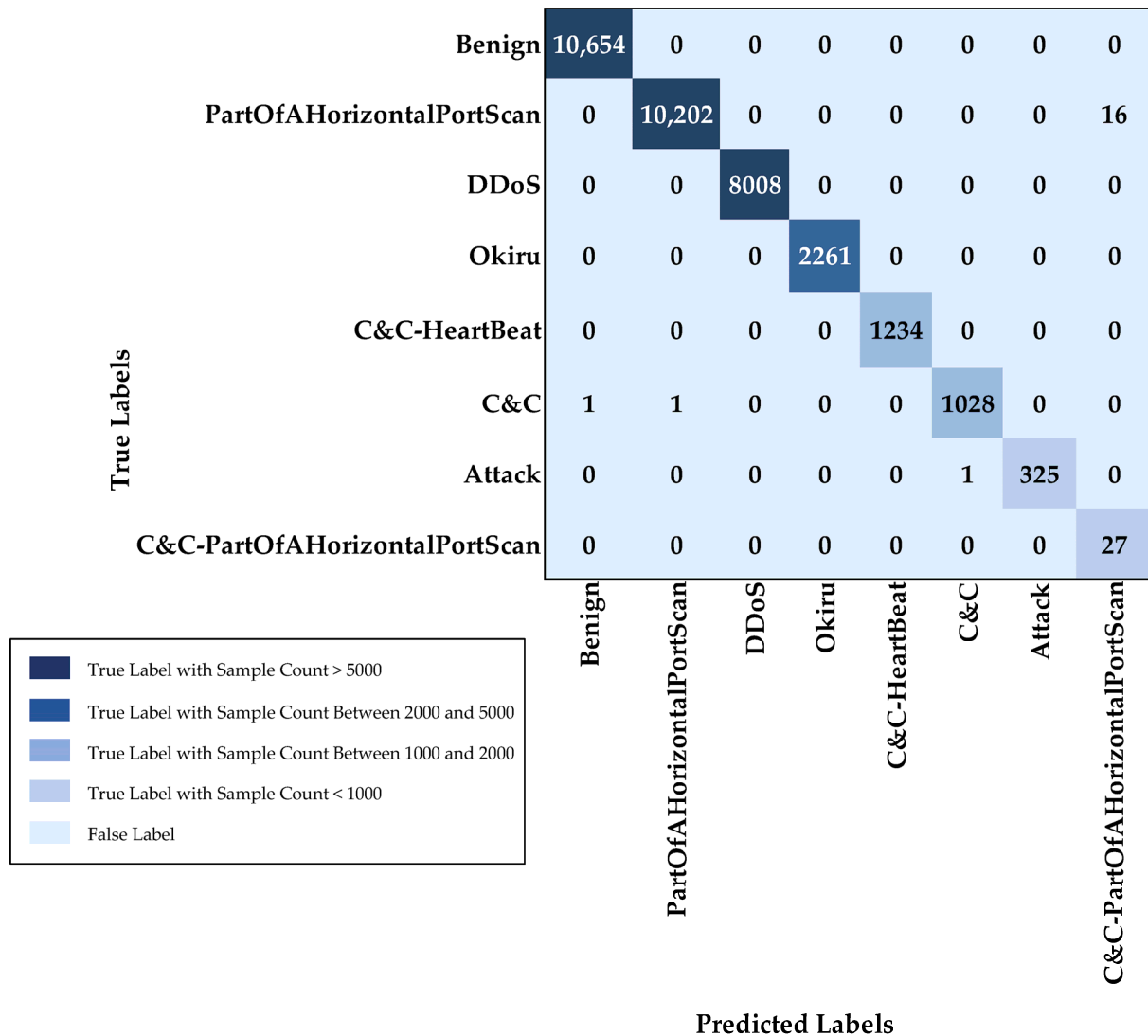


Figure 6. Confusion matrix for multi-class classification using the CNN-DNN model on IoT-23 dataset.

The multi-class confusion matrix for the combined dataset, assessed using the proposed CNN-DNN model, highlights its strong ability to differentiate among benign traffic and attack categories, as shown in Figure 7. Benign traffic is classified with high precision, with 778 instances correctly identified and 66 misclassified. Prominent DDoS attacks, including LOIC-HTTP and HOIC, are reliably detected, achieving 5024 and 4050 correct classifications, respectively. DoS attacks, such as Hulk and GoldenEye, are recognized with 4207 and 2045 correct identifications, while Bot and Infiltration attacks also demonstrate strong detection performance, with 3944 and 726 instances accurately classified, respectively. Additional attack classes, including SSH-Bruteforce, Brute Force-Web, SQL Injection, FTP-BruteForce, and Reconnaissance, are consistently distinguished with high performance, with correct counts ranging from 1 to 4846. Complex attacks, such as DDoS, DoS, Theft, PartOfAHorizontalPortScan, Okiru, C&C-HeartBeat, C&C, Attack, and C&C-PartOfAHorizontalPortScan, are effectively identified with correct classifications ranging from 17 to 20,199, reflecting the model's capability to handle sophisticated attack behaviors. Overall, the confusion matrix underscores the robustness, reliability, and precision of the proposed CNN-DNN model, confirming its exceptional performance in multi-class intrusion detection for the combined dataset.

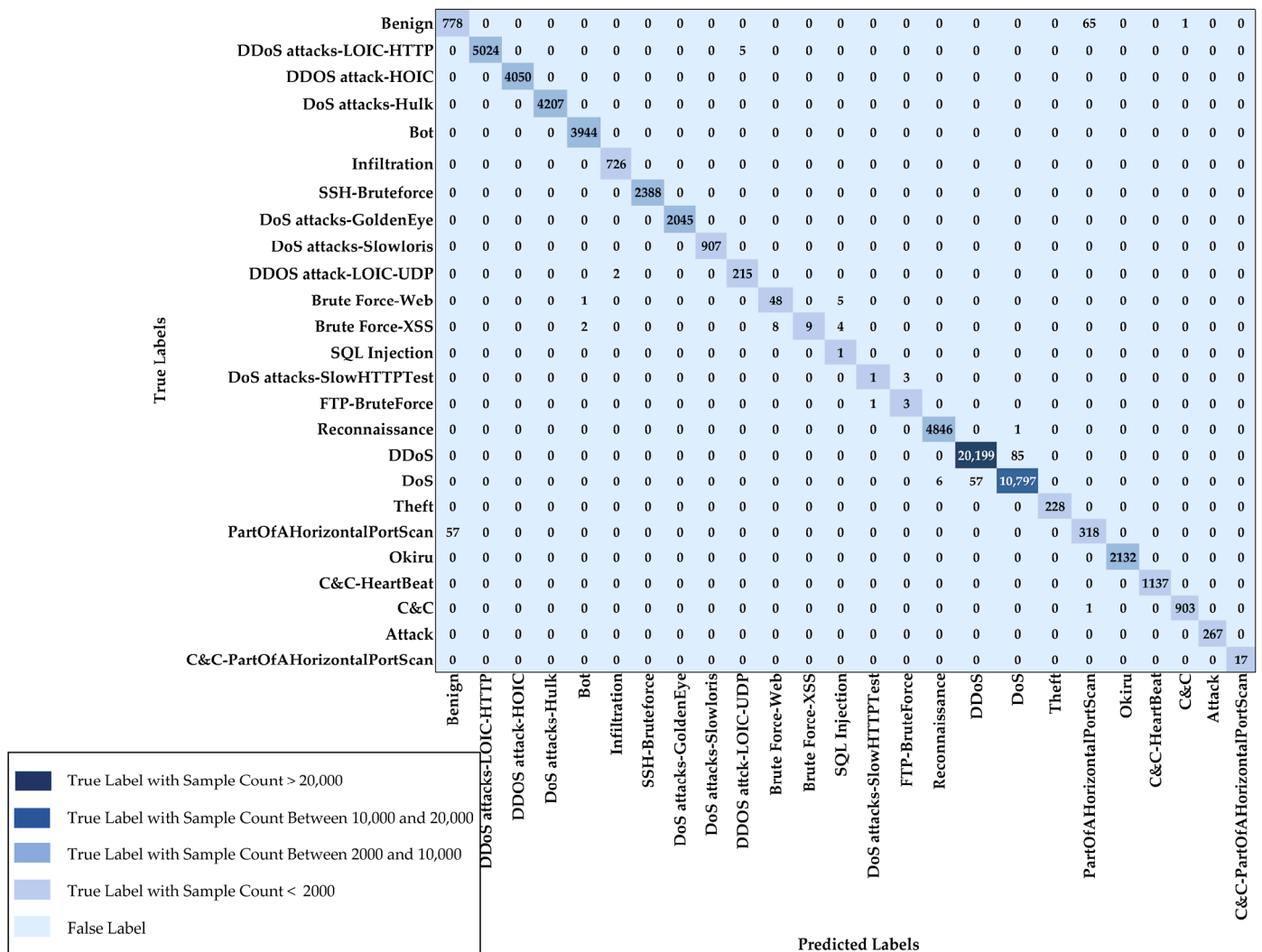


Figure 7. Confusion matrix for multi-class classification using the CNN-DNN model on combined dataset (using shared autoencoder).

The multi-class classification results provide compelling evidence of the enhanced effectiveness of the proposed CNN-DNN framework compared with benchmark models across all evaluated datasets, as shown in Figure 8. On the CSE-CIC-IDS2018 dataset, the hybrid architecture achieves the highest accuracy of 99.61%, slightly outperforming CNN at 99.58%, autoencoder at 99.54%, and the standalone DNN at 99.57%, demonstrating its superior ability to differentiate between diverse and closely related traffic types. For the NF-BoT-IoT-v2 dataset, where traffic patterns are more complex, the proposed CNN-DNN attains the best accuracy of 98.28%, exceeding CNN at 98.23%, DNN at 98.19%, and autoencoder at 98.21%. A consistent pattern of superiority is also observed on the IoT-23 dataset, where the proposed model records 99.94%, surpassing CNN at 99.91%, DNN at 99.75%, and autoencoder at 99.54%, confirming its robustness in multi-class classification. When evaluated on the combined dataset constructed using the shared autoencoder, the CNN-DNN again delivers the highest accuracy of 99.54%, outperforming CNN at 99.48%, DNN at 99.49%, and autoencoder at 99.46%. Collectively, these findings highlight the strong discriminative power, stability, and generalization capability of the CNN-DNN architecture in addressing complex multi-class intrusion detection scenarios across both individual and heterogeneous aggregated datasets.

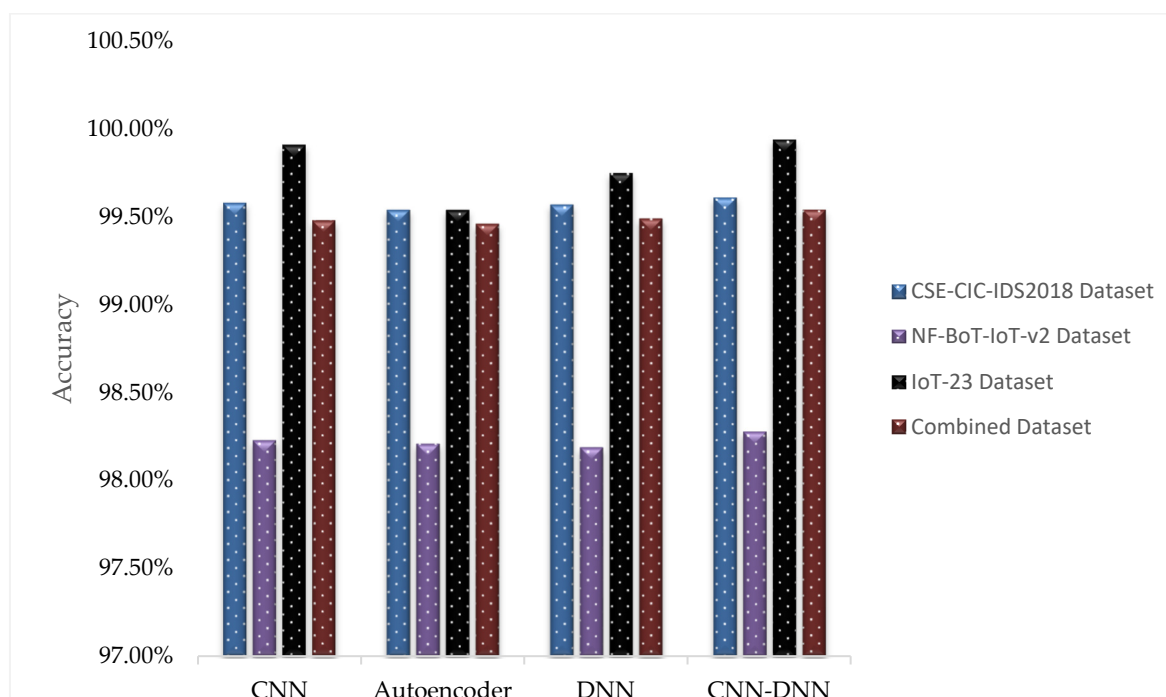


Figure 8. Proposed CNN–DNN versus multi-class classifiers.

The multi-class assessment of the CNN–DNN model on the CSE-CIC-IDS2018 dataset, as presented in Table 29, illustrates robust and consistent detection performance across most traffic categories. Several attack types, including DDOS attack-HOIC, SSH-Bruteforce, and DoS attacks-Hulk, achieve perfect scores in accuracy, precision, recall, and F1-score, highlighting the model's capability to effectively capture distinct patterns for these classes. Other significant attack categories also demonstrate excellent results, such as DDoS attacks-LOIC-HTTP with 99.92% accuracy, DoS attacks-GoldenEye at 99.97% accuracy, and DDOS attack-LOIC-UDP achieving 100% recall and 98.04% precision. Normal (benign) traffic is correctly identified with 97.17% accuracy, while Bot and Infiltration classes reach 100% and 93.57% accuracy, respectively, further showcasing the model's ability to distinguish between legitimate behavior and varied malicious activities. The findings confirm the CNN–DNN model's effectiveness and strong generalization capacity in handling complex multi-class intrusion detection tasks.

Table 29. Multi-class classification performance metrics for each class using the CNN–DNN model on the CSE-CIC-IDS2018 dataset.

Class	Accuracy	Precision	Recall	F-Score
Benign	97.17%	91.39%	97.17%	94.19%
DDoS attacks-LOIC-HTTP	99.92%	100%	99.92%	99.96%
DDOS attack-HOIC	100%	100%	100%	100%
DoS attacks-Hulk	100%	100%	100%	100%
Bot	100%	99.89%	100%	99.95%
Infiltration	93.57%	97.94%	93.57%	95.70%
SSH-Bruteforce	100%	100%	100%	100%
DoS attacks-GoldenEye	99.97%	100%	99.97%	99.99%
DoS attacks-Slowloris	100%	99.94%	100%	99.97%
DDOS attack-LOIC-UDP	100%	98.04%	100%	99.01%
Brute Force-Web	96.08%	84.48%	96.08%	89.91%
Brute Force-XSS	40.74%	100%	40.74%	57.89%
SQL Injection	33.33%	100%	33.33%	50%

DoS attacks-SlowHTTPTest	33.33%	50%	33.33%	40%
FTP-BruteForce	66.67%	50%	66.67%	57.14%

An examination of the per-class performance of the CNN–DNN model for multi-class classification on the NF-BoT-IoT-v2 and IoT-23 datasets is presented in Tables 30 and 31. On the NF-BoT-IoT-v2 dataset, the model demonstrates excellent detection capability across all traffic categories, achieving 100% accuracy for the Theft class, with recall reaching 100% and precision and F1-score remaining very high. Strong performance is also observed for Benign, Reconnaissance, DDoS, and DoS traffic, with accuracies of 99.94%, 95.85%, 99.37%, and 98.12%, respectively. For the IoT-23 dataset, most classes, including Benign, PartOfAHorizontalPortScan, C&C, and Attack, achieve consistently high performance across all evaluation metrics. In addition, the model attains perfect performance (100% accuracy, precision, recall, and F1-score) for the DDoS, Okiru, and C&C-HeartBeat classes. Overall, these results highlight the CNN–DNN model's effectiveness and its ability to accurately classify a wide range of benign and malicious traffic types across heterogeneous environments.

Table 30. Multi-class classification performance metrics for each class using the CNN–DNN model on the NF-BoT-IoT-v2 dataset.

Class	Accuracy	Precision	Recall	F-Score
Benign	99.94%	99.94%	99.94%	99.94%
Reconnaissance	95.85%	98.40%	95.85%	97.11%
DDoS	99.37%	99.41%	99.37%	99.39%
DoS	98.12%	96.19%	98.12%	97.15%
Theft	100%	97.98%	100%	98.98%

Table 31. Multi-class classification performance metrics for each class using the CNN–DNN model on the IoT-23 dataset.

Class	Accuracy	Precision	Recall	F-Score
Benign	100%	99.99%	100%	100%
PartOfAHorizontalPortScan	99.84%	99.99%	99.84%	99.92%
DDoS	100%	100%	100%	100%
Okiru	100%	100%	100%	100%
C&C-HeartBeat	100%	100%	100%	100%
C&C	99.81%	99.90%	99.81%	99.85%
Attack	99.69%	100%	99.69%	99.85%
C&C-PartOfAHorizontalPortScan	100%	62.79%	100%	77.14%

The multi-class classification results obtained by the CNN–DNN model, as presented in Table 32, demonstrate strong and reliable detection performance across most classes in the combined dataset. Several attack categories achieve perfect classification with 100% accuracy, recall, and F-score, including DDOS attack-HOIC, DoS attacks-Hulk, SSH-Bruteforce, DoS attacks-GoldenEye, DoS attacks-Slowloris, Theft, Okiru, C&C-HeartBeat, Attack, and C&C-PartOfAHorizontalPortScan, highlighting the model's strong learning capability. High performance is also maintained for aggregated attack groups such as DDoS and DoS, which achieve accuracies of 99.58% and 99.42%, respectively. Reconnaissance traffic is classified with 99.98% accuracy, accompanied by consistently high precision and recall. Benign traffic is identified with 92.18% accuracy and 93.17% precision, indicating effective differentiation between normal and malicious behavior despite in-

creased class diversity in the combined dataset. Overall, the results confirm that the hybrid CNN–DNN architecture, combined with the shared autoencoder representation, provides robust generalization and effective multi-class intrusion detection.

Table 32. Multi-class classification performance metrics for each class using the CNN–DNN model on the combined dataset (using shared autoencoder).

Class	Accuracy	Precision	Recall	F-Score
Benign	92.18%	93.17%	92.18%	92.67%
DDoS attacks-LOIC-HTTP	99.90%	100%	99.90%	99.95%
DDoS attack-HOIC	100%	100%	100%	100%
DoS attacks-Hulk	100%	100%	100%	100%
Bot	100%	99.92%	100%	99.96%
Infiltration	100%	99.73%	100%	99.86%
SSH-Bruteforce	100%	100%	100%	100%
DoS attacks-GoldenEye	100%	100%	100%	100%
DoS attacks-Slowloris	100%	100%	100%	100%
DDoS attack-LOIC-UDP	99.08%	97.73%	99.08%	98.40%
Brute Force-Web	88.89%	85.71%	88.89%	87.27%
Brute Force-XSS	39.13%	100%	39.13%	56.25%
SQL Injection	100%	10%	100%	18.18%
DoS attacks-SlowHTTPTest	25%	50%	25%	33.33%
FTP-BruteForce	75%	50%	75%	60%
Reconnaissance	99.98%	99.88%	99.98%	99.93%
DDoS	99.58%	99.72%	99.58%	99.65%
DoS	99.42%	99.21%	99.42%	99.31%
Theft	100%	100%	100%	100%
PartOfAHorizontalPortScan	84.80%	82.81%	84.80%	83.79%
Okiru	100%	100%	100%	100%
C&C-HeartBeat	100%	100%	100%	100%
C&C	99.89%	99.89%	99.89%	99.89%
Attack	100%	100%	100%	100%
C&C-PartOfAHorizontalPortScan	100%	100%	100%	100%

6. Limitations

Our method overcomes important drawbacks including increased attack coverage, high performance, complex combination architecture, generalization, and scalability, yet certain issues still exist:

- **Combination of Multiple Datasets:** While the current framework effectively integrates three heterogeneous datasets, extending the approach to include more datasets could introduce extra challenges in preprocessing, integration, and training, necessitating careful design to preserve information and maintain reconstruction fidelity.
- **Data Preprocessing:** The quality of the preprocessed data has a direct impact on the model's performance. Accurately handling missing values, properly encoding category data, and successfully normalizing numerical data are all essential.
- **Model Adaptation:** Hyperparameter tuning calls for a thorough and iterative process in order to optimize the model for optimal performance on a variety of datasets. This procedure is essential in order to guarantee that the model precisely matches the unique characteristics of each new dataset.

7. Conclusions

This paper introduces a unified intrusion detection framework that implements “Structural Dualism” to integrate three heterogeneous benchmark datasets (CSE-CIC-IDS2018, NF-BoT-IoT-v2, and IoT-23) into a harmonized, protocol-agnostic representation. By utilizing a shared autoencoder with dataset-specific projection layers, the framework establishes a unified latent manifold. This 15-dimensional space successfully captures the underlying semantics of attack patterns across disparate domains, while dataset-specific decoders maintain reconstruction fidelity through alternating multi-domain training. To achieve high-fidelity classification within this manifold, a synergistic hybrid CNN–DNN architecture was developed. In this design, the CNN layers are specifically engineered to extract spatial micro-signatures from the latent vector, while the DNN performs global relational mapping across twenty-five distinct traffic classes. Experimental results on the merged latent dataset demonstrate outstanding performance, achieving 99.76% accuracy for binary classification and 99.54% for multi-class tasks. Furthermore, individual dataset evaluations confirm the framework’s superior generalization. These results highlight the framework’s ability to move beyond superficial statistical noise to recognize universal intrusion behaviors. Overall, the shared autoencoder-based CNN–DNN framework provides an effective, scalable, and resilient solution for safeguarding complex, heterogeneous network environments.

8. Future Work

Future studies have to concentrate on the restrictions and difficulties highlighted in Section 6:

- **Combination of Multiple Datasets:** Future work will explore the extension of the framework to integrate more than three heterogeneous datasets, investigating scalable methods for data integration, harmonization, and model training. This includes developing strategies to maintain information integrity, reconstruction fidelity, and high detection performance when combining a larger and more diverse set of network traffic sources.
- **Data Preprocessing:** Customizing data preparation techniques for every dataset is essential to achieving optimal model performance. Refer to Section 3.2 in this article for a more thorough explanation of these sophisticated methods.
- **Hyperparameter Optimization and Model Adaptation:** More advanced hyperparameter optimization techniques are essential for deployment success in a variety of data environments. Section 3.3 goes into depth about these tuning techniques.

Author Contributions: Conceptualization, H.K. and M.M.; Methodology, H.K. and M.M.; Software, H.K. and M.M.; Validation, H.K. and M.M.; Formal analysis, H.K. and M.M.; Investigation, H.K. and M.M.; Resources, H.K. and M.M.; Data curation, H.K. and M.M.; Writing—original draft, H.K. and M.M.; Writing—review & editing, H.K. and M.M.; Visualization, H.K. and M.M.; Supervision, M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used in our study, CSE-CIC-IDS2018, NF-BoT-IoT-v2 and IoT-23, are publicly available. Below are the URLs for the datasets: CSE-CIC-IDS2018: <https://www.unb.ca/cic/datasets/ids-2018.html> (accessed on 9 February 2026). NF-BoT-IoT-v2: https://staff.itee.uq.edu.au/marius/NIDS_datasets/ (accessed on 9 February 2026). IoT-23: <https://zenodo.org/records/4743746> (accessed on 9 February 2026).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sun, P.; Liu, P.; Li, Q.; Liu, C.; Lu, X.; Hao, R.; Chen, J. DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System. *Secur. Commun. Netw.* **2020**, *2020*, 8890306.
2. Almansor, M.; Gan, K. Intrusion detection systems: Principles and perspectives. *J. Multidiscip. Eng. Sci. Stud.* **2018**, *4*, 2458–2925.
3. Alkahtani, H.; Aldhyani, T.H. Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms. *Complexity* **2021**, *2021*, 5579851.
4. Edeh, D.I. Network Intrusion Detection System Using Deep Learning Technique. Master's Thesis, Department of Computing, University of Turku, Turku, Finland, 2021.
5. Wu, P. Deep Learning for Network Intrusion Detection: Attack Recognition with Computational Intelligence. Master's Thesis, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia, 2020.
6. Altulaihan, E.; Almaiah, M.A.; Aljughaiman, A. Anomaly detection IDS for detecting DoS attacks in IoT networks based on machine learning algorithms. *Sensors* **2024**, *24*, 713.
7. Zhang, D.; Huang, D.; Chen, Y.; Lin, S.; Li, C. A lightweight IoT intrusion detection method based on two-stage feature selection and Bayesian optimization. *AIMS Electron. Electr. Eng.* **2025**, *9*, 359–389.
8. Gad, A.R.; Haggag, M.; Nashat, A.A.; Barakat, T.M. A distributed intrusion detection system using machine learning for IoT based on ToN-IoT dataset. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 548–563.
9. Alsharif, N.A.; Mishra, S.; Alshehri, M. IDS in IoT using Machine Learning and Blockchain. *Eng. Technol. Appl. Sci. Res.* **2023**, *13*, 11197–11203.
10. Hakami, H.; Faheem, M.; Ahmad, M.B. Machine Learning Techniques for Enhanced Intrusion Detection in IoT Security. *IEEE Access* **2025**, *13*, 31140–31158.
11. ElKashlan, M.; Elsayed, M.S.; Jurcut, A.D.; Azer, M. A machine learning-based intrusion detection system for iot electric vehicle charging stations (evcss). *Electronics* **2023**, *12*, 1044.
12. Abdelkhalek, A.; Mashaly, M. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. *J. Supercomput.* **2023**, *79*, 10611–10644.
13. Ajdani, M. Deep Learning-Based Intrusion Detection Systems: A Novel Approach Using Generative Adversarial Networks (GANs). *Int. J. Inf. Secur. Priv.* **2025**, *19*, 1–5.
14. Saba, T.; Rehman, A.; Sadad, T.; Kolivand, H.; Bahaj, S.A. Anomaly-based intrusion detection system for IoT networks through deep learning model. *Comput. Electr. Eng.* **2022**, *99*, 107810.
15. Alghamdi, R.; Bellaiche, M. An ensemble deep learning based IDS for IoT using Lambda architecture. *Cybersecurity* **2023**, *6*, 5.
16. Aljuaid, W.H.; Alshamrani, S.S. A deep learning approach for intrusion detection systems in cloud computing environments. *Appl. Sci.* **2024**, *14*, 5381.
17. Nawaz, M.W.; Munawar, R.; Mehmood, A.; Rahman, M.M.U.; Qammer; Abbasi, H. Multi-class Network Intrusion Detection with Class Imbalance via LSTM & SMOTE. *arXiv* **2023**, arXiv:2310.01850.
18. Xi, C.; Wang, H.; Wang, X. A novel multi-scale network intrusion detection model with transformer. *Sci. Rep.* **2024**, *14*, 23239.
19. Shu, L.; Dong, S.; Su, H.; Huang, J. Android malware detection methods based on convolutional neural network: A survey. *IEEE Trans. Emerg. Top. Comput. Intell.* **2023**, *7*, 1330–1350.
20. Li, J.; Zhou, H.; Wu, S.; Luo, X.; Wang, T.; Zhan, X.; Ma, X. FOAP: Fine-Grained Open-World Android App Fingerprinting. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 2022)*; USENIX Association: Boston, MA, USA, 10–12 August; pp. 1579–1596.
21. Ni, T.; Lan, G.; Wang, J.; Zhao, Q.; Xu, W. Eavesdropping Mobile App Activity via Radio-Frequency Energy Harvesting. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 2023)*; USENIX Association: Santa Clara, CA, USA, 9–11 August 2023; pp. 3511–3528.
22. Kamal, H.; Mashaly, M. AE-DTNN: Autoencoder-Dense-Transformer Neural Network Model for Efficient Anomaly-Based Intrusion Detection Systems. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 78.
23. Fu, Y.; Du, Y.; Cao, Z.; Li, Q.; Xiang, W. A deep learning model for network intrusion detection with imbalanced data. *Electronics* **2022**, *11*, 898.
24. Qazi, E.U.; Faheem, M.H.; Zia, T. HDLNIDS: Hybrid deep-learning-based network intrusion detection system. *Appl. Sci.* **2023**, *13*, 4921.
25. Sajid, M.; Malik, K.R.; Almogren, A.; Malik, T.S.; Khan, A.H.; Tanveer, J.; Rehman, A.U. Enhancing intrusion detection: A hybrid machine and deep learning approach. *J. Cloud Comput.* **2024**, *13*, 123.

26. Kamal, H.; Mashaly, M. Robust Intrusion Detection System Using an Improved Hybrid Deep Learning Model for Binary and Multi-Class Classification in IoT Networks. *Technologies* **2025**, *13*, 102.
27. Umair, M.B.; Iqbal, Z.; Faraz, M.A.; Khan, M.A.; Zhang, Y.-D.; Razmjoo, N.; Kadry, S. A network intrusion detection system using hybrid multilayer deep learning model. *Big Data* **2024**, *12*, 367–376.
28. Kamal, H.; Mashaly, M. Enhanced Hybrid Deep Learning Models-Based Anomaly Detection Method for Two-Stage Binary and Multi-Class Classification of Attacks in Intrusion Detection Systems. *Algorithms* **2025**, *18*, 69.
29. Yaras, S.; Dener, M. IoT-based intrusion detection system using new hybrid deep learning algorithm. *Electronics* **2024**, *13*, 1053.
30. Kamal, H.; Mashaly, M. Advanced hybrid transformer-CNN deep learning model for effective intrusion detection systems with class imbalance mitigation using resampling techniques. *Future Internet* **2024**, *16*, 481.
31. Zheng, Z.X.; Chen, F. An IoT Intrusion Detection Method Combining GAN and Transformer Neural Networks. *J. Netw. Intell.* **2025**, *10*, 1011–1026.
32. Balaji, S.; Dhanabalan, G.; Umarani, C.; Naskath, J. A GAN-based Hybrid Deep Learning Approach for Enhancing Intrusion Detection in IoT Networks. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 1104–1112.
33. Afraji, D.M.; Lloret, J.; Peñalver, L. An integrated hybrid deep learning framework for intrusion detection in IoT and IIoT networks using CNN-LSTM-GRU architecture. *Computation* **2025**, *13*, 222.
34. Adefemi, K.O.; Mutanga, M.B.; Alimi, O.A. A Hybrid CNN-GRU Deep Learning Model for IoT Network Intrusion Detection. *J. Sens. Actuator Netw.* **2025**, *14*, 96.
35. Hasan, M.A.; Sharma, D. CNN-LSTM Powered Network IDS for Adaptive Cyber Defence. *Revolut. Adv. Comput. Electron. Int. J.* **2025**, *1*, 1–16.
36. Sun, Z.; Ni, T.; Yang, H.; Liu, K.; Zhang, Y.; Gu, T.; Xu, W. FLoRa: FLoRa: Energy-efficient, reliable, and beamforming-assisted over-the-air firmware update in lora networks. In *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks (IPSN '23)*; Association for Computing Machinery: New York, NY, USA, 9–12 May 2023; pp. 14–26.
37. Wang, J.; Ni, T.; Lee, W.B.; Zhao, Q. A Contemporary Survey of Large Language Model Assisted Program Analysis. *Trans. Artif. Intell.* **2025**, *1*, 105–129.
38. Wang, M.; Yang, N.; Guo, Y.; Weng, N. Learn-ids: Bridging gaps between datasets and learning-based network intrusion detection. *Electronics* **2024**, *13*, 1072.
39. Al-Riyami, S.; Lisitsa, A.; Coenen, F. Cross-datasets evaluation of machine learning models for intrusion detection systems. In *Proceedings of Sixth International Congress on Information and Communication Technology: ICICT 2021, London, UK, 27 October 2021*; Springer Singapore: Singapore, 2021; Volume 4, pp. 815–828.
40. Yu, J.; Wang, G.; Shi, N.; Saxena, R.; Lee, B. A Multi-View-Based Federated Learning Approach for Intrusion Detection. *Electronics* **2025**, *14*, 4166.
41. Alabbadi, A.; Bajaber, F. X-FuseRLSTM: A Cross-Domain Explainable Intrusion Detection Framework in IoT Using the Attention-Guided Dual-Path Feature Fusion and Residual LSTM. *Sensors* **2025**, *25*, 3693.
42. Amin, M.I.; Shen, M.; Ishak, M.K.; Manickam, S.; Karuppayah, S. Enhancing generalization of cross-domain intrusion detection: A heterogeneous deep stacked ensemble approach. *Connect. Sci.* **2026**, *38*, 2599708.
43. Li, M.; Qiao, Y.; Lee, B. Multi-View Intrusion Detection Framework Using Deep Learning and Knowledge Graphs. *Information* **2025**, *16*, 377.
44. Kamal, H.; Mashaly, M. Combined Dataset System Based on a Hybrid PCA-Transformer Model for Effective Intrusion Detection Systems. *AI* **2025**, *6*, 168.
45. Alghamdi, R.; Bellaiche, M. Evaluation and selection models for ensemble intrusion detection systems in IoT. *IoT* **2022**, *3*, 285–314.
46. Alabdulwahab, S.; Kim, Y.T.; Seo, A.; Son, Y. Generating synthetic dataset for ML-based IDS using CTGAN and feature selection to protect smart IoT environments. *Appl. Sci.* **2023**, *13*, 10951.
47. Elouardi, S.; Motii, A.; Jouhari, M.; Amadou, A.N.; Hedabou, M. A survey on Hybrid-CNN and LLMs for intrusion detection systems: Recent IoT datasets. *IEEE Access* **2024**, *12*, 180009–180033.
48. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. “CSE-CIC-IDS2018 Dataset.” Canadian Institute for Cybersecurity, University of New Brunswick. 2018. Available online: <https://www.unb.ca/cic/datasets/ids-2018.html> (accessed on 9 February 2026).
49. Songma, S.; Sathuphan, T.; Pamutha, T. Optimizing intrusion detection systems in three phases on the 2053 CSE-CIC-IDS-2018 dataset. *Computers* **2023**, *12*, 245.
50. Sarhan, M.; Layeghy, S.; Portmann, M. Towards a standard feature set for network intrusion detection system datasets. *Mob. Netw. Appl.* **2022**, *27*, 357–370.

51. Moustafa, N. Network Intrusion Detection System (NIDS) Datasets. University of Queensland: Brisbane, Australia. Available online: https://staff.itee.uq.edu.au/marius/NIDS_datasets (accessed on 21 June 2025).
52. Balaji, R.; Deepajothi, S.; Prabakaran, G.; Daniya, T.; Karthikeyan, P.; Velliangiri, S. Survey on intrusions detection system using deep learning in iot environment. In Proceedings of the 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 7–9 April 2022; pp. 195–199.
53. Garcia, S.; Parmisano, A.; Erquiaga, M.J. IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic. Zenodo. 2021. Available online: <https://zenodo.org/records/4743746> (accessed on 20 February 2024).
54. Abdalgawad, N.; Sajun, A.; Kaddoura, Y.; Zualkernan, I.A.; Aloul, F. Generative deep learning to detect cyberattacks for the IoT-23 dataset. *IEEE Access* **2021**, *10*, 6430–6441.
55. Kim, Y.G.; Ahmed, K.J.; Lee, M.J.; Tsukamoto, K. A Comprehensive Analysis of Machine Learning-Based Intrusion Detection System for IoT-23 Dataset. In *International Conference on Intelligent Networking and Collaborative Systems*; Springer International Publishing: Cham, Switzerland, 2022; pp. 475–486.
56. Kotsiantis, S.B.; Kanellopoulos, D.; Pintelas, P.E. Data preprocessing for supervised learning. *Int. J. Comput. Sci.* **2006**, *1*, 111–117.
57. Han, J.; Pei, J.; Tong, H. *Data Mining: Concepts and techniques*; Morgan Kaufmann: Burlington, MA, USA, 2022.
58. Little, R.J.; Rubin, D.B. *Statistical Analysis with Missing Data*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
59. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
60. Singh, D.; Singh, B. Investigating the impact of data normalization on classification performance. *Appl. Soft Comput.* **2020**, *97*, 105524.
61. Sola, J.; Sevilla, J. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Trans. Nucl. Sci.* **1997**, *44*, 1464–1468.
62. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359.
63. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011), Fort Lauderdale, FL, USA, 11–13 April 2011.
64. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
65. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507.
66. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828.
67. Tishby, N.; Zaslavsky, N. Deep learning and the information bottleneck principle. In Proceedings of the 2015 IEEE Information Theory Workshop (ITW 2015), Jerusalem, Israel, 26 April–1 May 2015.
68. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-baiot—Network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22.
69. Bousmalis, K.; Trigeorgis, G.; Silberman, N.; Krishnan, D.; Erhan, D. Domain separation networks. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Volume 29.
70. Caruana, R. Multitask learning. *Mach. Learn.* **1997**, *28*, 41–75.
71. Kendall, A.; Gal, Y.; Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7482–7491.
72. Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv* **2017**, arXiv:1706.05098.
73. Ganin, Y.; Lempitsky, V. Unsupervised domain adaptation by backpropagation. In Proceedings of the 32nd International Conference on Machine Learning (ICML 2015), Lille, France, 6–11 July 2015; pp. 1180–1189.
74. Rizzardi, A.; Sicari, S.; Coen-Porisini, A. HERO: From High-dimensional network traffic to zERO-Day attack detection. *Comput. Netw.* **2025**, *265*, 111264.
75. Rizzardi, A.; Sicari, S.; Porisini, A.C. NERO: NEural algorithmic reasoning for zeRO-day attack detection in the IoT: A hybrid approach. *Comput. Secur.* **2024**, *142*, 103898.
76. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the International Conference on Information Systems Security and Privacy, Funchal, Portugal, 22–24 January 2018; pp. 108–116.
77. Nair Vinod Geoffrey, E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel, 21–24 June 2010; pp. 807–814.

78. Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
79. Nielsen, M.A. *Neural Networks and Deep Learning*; Determination Press: San Francisco, CA, USA, 2015.
80. Bishop, C.M.; Nasser, M.N. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006; Volume 4, no. 4.
81. El-Habil, B.Y.; Abu-Naser, S.S. Global climate prediction using deep learning. *J. Theor. Appl. Inf. Technol.* **2022**, *100*, 4824–4838.
82. Song, Z.; Ma, J. Deep learning-driven MIMO: Data encoding and processing mechanism. *Phys Commun.* **2022**, *57*, 101976.
83. Zhou, X.; Zhao, C.; Sun, J.; Yao, K.; Xu, M. Detection of lead content in oilseed rape leaves and roots based on deep transfer learning and hyperspectral imaging technology. *Spectroch. Acta Part A Mol. Biomol. Spectrosc.* **2022**, *290*, 122288.
84. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **2002**, *86*, 2278–2324.
85. Kunang, Y.N.; Nurmaini, S.; Stiawan, D.; Zarkasi, A. Automatic features extraction using autoencoder in intrusion detection system. In Proceedings of the 2018 International Conference on Electrical Engineering and Computer Science (ICECOS), Pangkal Pinang, Indonesia, 2–4 October 2018; pp. 219–224.
86. Gogna, A.; Majumdar, A. Discriminative autoencoder for feature extraction: Application to character recognition. *Neural Processing Lett.* **2019**, *49*, 1723–1735.
87. Chen, X.; Ma, L.; Yang, X. Stacked denoise autoencoder based feature extraction and classification for hyperspectral images. *J. Sens.* **2016**, *2016*, 3632943.
88. Michelucci, U. An introduction to autoencoders. *arXiv* **2022**, arXiv:2201.03898.
89. Veeramreddy, J.; Prasad, K. Anomaly-Based Intrusion Detection System. In *Anomaly Detection and Complex Network Systems*; Alexandrov, A.A., Ed.; IntechOpen: London, UK, 2019.
90. Chen, C.; Song, Y.; Yue, S.; Xu, X.; Zhou, L.; Lv, Q.; Yang, L. FCNN-SE: An Intrusion Detection Model Based on a Fusion CNN and Stacked Ensemble. *Appl. Sci.* **2022**, *12*, 8601.
91. Powers, D.M.W. Evaluation: From Precision, Recall, and F-Measure to ROC, Informedness, Markedness & Correlation. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
92. Assy, A.T.; Mostafa, Y.; El-Khaleq, A.A.; Mashaly, M. Anomaly-based intrusion detection system using one-dimensional convolutional neural network. *Procedia Comput. Sci.* **2023**, *220*, 78–85.
93. Kamal, H.; Mashaly, M. Hybrid Deep Learning-Based Autoencoder-DNN Model for Intelligent Intrusion Detection System in IoT Networks. In Proceedings of the 2025 15th International Conference on Electrical Engineering (ICEENG), Cairo, Egypt, 12–15 May 2025; pp. 1–6.
94. Kamal, H.; Mashaly, M. Improving Anomaly Detection in IDS with Hybrid Auto Encoder-SVM and Auto Encoder-LSTM Models Using Resampling Methods. In Proceedings of the 2024 6th Novel Intelligent and Leading Emerging Sciences Conference (NILES), Giza, Egypt, 19–21 October 2024; pp. 34–39.
95. Kamal, H.; Mashaly, M. Securing IIoT Networks Using a Hybrid PCA-CNN-Based Intrusion Detection System. In *2025 7th Novel Intelligent and Leading Emerging Sciences Conference (NILES), Giza, Egypt, 25–27 October 2025*; IEEE: New York, NY, USA, 2025; pp. 19–24.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.