DTL-5G: Deep Transfer Learning-based DDoS Attack Detection in 5G and Beyond Networks

Behnam Farzaneh[a], Nashid Shahriar[a], Abu Hena Al Muktadir[b], Md. Shamim Towhid[a], Mohammad Sadegh Khosravani[a]

[a]*University of Regina, Department of Computer Science, Regina, Canada*
[b]*Saskatchewan Polytechnic, Department of Computer Science, Regina, Canada*

Abstract

Network slicing is considered as a key enabler for 5G and beyond mobile networks for supporting a variety of new services, including enhanced mobile broadband, ultra-reliable and low-latency communication, and massive connectivity, on the same physical infrastructure. However, this technology increases the susceptibility of networks to cyber threats, particularly Distributed Denial-of-Service (DDoS) attacks. These attacks have the potential to cause service quality degradation by overloading network function(s) that are central to network slices to operate seamlessly. This calls for an Intrusion Detection System (IDS) as a shield against a wide array of DDoS attacks. In this regard, one promising solution would be the use of Deep Learning (DL) models for detecting possible DDoS attacks, an approach that has already made its way into the field given its manifest effectiveness. However, one particular challenge with DL models is that they require large volumes of labeled data for efficient training, which are not readily available in operational networks. A possible workaround is to resort to Transfer Learning (TL) approaches that can utilize the knowledge learned from prior training to a target domain with limited labeled data. This paper investigates how Deep Transfer Learning (DTL) based approaches can improve the detection of DDoS attacks in 5G networks by leveraging DL models, such as Bidirectional Long Short-

*Corresponding author: Behnam Farzaneh
Email address: bfd654@uregina.ca, nashid.shahriar@uregina.ca, muktadir.uec@gmail.com, mty754@uregina.ca, mko041@uregina.ca (Behnam Farzaneh[a], Nashid Shahriar[a], Abu Hena Al Muktadir[b], Md. Shamim Towhid[a], Mohammad Sadegh Khosravani[a])

Term Memory (BiLSTM), Convolutional Neural Network (CNN), Residual Network (ResNet), and Inception as base models. A comprehensive dataset generated in our 5G network slicing testbed serves as the source dataset for DTL, which includes both benign and different types of DDoS attack traffic. After learning features, patterns, and representations from the source dataset using initial training, we fine-tune base models using a variety of TL processes on a target DDoS attack dataset. The 5G-NIDD dataset, which has a sparse amount of annotated traffic pertaining to several DDoS attack generated in a real 5G network, is chosen as the target dataset. The results show that the proposed DTL models have performance improvements in detecting different types of DDoS attacks in 5G-NIDD dataset compared to the case when no TL is applied. According to the results, the BiLSTM and Inception models being identified as the top-performing models. BiLSTM indicates an improvement of 13.90%, 21.48%, and 12.22% in terms of accuracy, recall, and F1-score, respectively, whereas, Inception demonstrates an enhancement of 10.09% in terms of precision, compared to the models that do not adopt TL.

Keywords:
5G Networks, Network Slicing, Intrusion Detection, DDoS Attacks, Deep Transfer Learning, Fine-tuning

---

1. Introduction

5G and Beyond 5G (B5G) mobile networks are expected to support a variety of emerging use-cases, such as holographic telepresence, immersive extended-reality, control and automation procedures in Industry 4.0, autonomous vehicles, Internet of Things (IoT), and remote telehealth [1]. Support for such a diverse set of applications imposes strict requirements on the mobile network along several dimensions, such as throughput, latency, and reliability. Network slicing has been envisaged as a key enabler to satisfy these diverse requirements, by creating multiple isolated end-to-end virtual networks dedicated to different services, on top of a common physical infrastructure. Some of the fundamental principles involved in realizing end-to-end network slicing are Network Function Virtualization and Software Defined Networking (SDN), which provide the necessary flexibility to tailor the network according to specific requirements. To achieve end-to-end slicing, the principles of NFV and SDN must be applied across the entirety of the mobile

network, from the Radio Access Network (RAN) to the core. Nevertheless, privacy and security concerns arising from the introduction of heterogeneous types of devices, adoption of new technologies such as SDN and NFV, and architectural changes, have eclipsed the advantages of the approaches mentioned above [2].

Similar to any other network infrastructure, 5G and B5G mobile networks are vulnerable to cyberthreats, particularly DoS/DDoS attacks [3]. In fact, the distributed deployment of network functions enabled by NFV and SDN increases the number of attack surfaces that can be made target for DDoS attacks. On the other hand, the scale and heterogeneity of User Equipments (UEs), such as smartphones and Internet of Things (IoT) devices, make them susceptible to be compromised easily and to be part of botnets that can be utilized to orchestrate DDoS attacks. A DDoS attack exhausts the network's resources by inundating it with illegitimate and bogus traffic from the attacker. The primary objective is typically to impede the network resource from fulfilling and responding to requests from authorized users [2, 4]. Differentiating between attack-generated and legitimate user traffic is necessary to mitigate such DDoS attacks in 5G and B5G mobile networks [2].

Detecting DDoS attacks in 5G and B5G mobile networks is an intricate task due to ever-changing network structures and the large-scale of network-connected devices that produce an immense traffic load on these networks. Furthermore, conventional IDSs face a deluge of increasingly complex attacks that outpace their defensive efforts. Data-driven solutions such as DL models have shown promise in dealing with voluminous traffic load and recognizing complex attack patterns to detect possible DDoS attacks. Traditional DL-based DDoS attack detection models are usually trained on a specific dataset originated from a particular network. DL models developed by training on a dataset from one network cannot be directly applied on another network due to change in data distribution and pattern. Each network needs to go through the same cycle of training a DL model from scratch for its own data that is often a time-consuming and tedious task. This challenge is further exacerbated by the scarcity of the required amount of labeled data needed for generating efficient DL models without any over- and under-fitting. Transfer Learning (TL) especially Deep Transfer Learning (DTL) strategies have been successfully used in computer vision and Natural Language Processing (NLP) to address similar challenges [5, 6].

DTL approaches involve repurposing a DL model developed for one domain—referred to as the source domain—to address a different but related

task within another domain, known as the target domain [5]. Rather than initiating a model's development from scratch for each new domain, DTL utilizes the knowledge and patterns learned from the source domain to improve model performance in the target domain. This is particularly beneficial when there is a shortage of data in the target domain and the information collected from the source domain are relevant to the target domain. Additionally, DTL can expedite model convergence and mitigate the resources needed to develop a custom-built DDoS attack detection models customized for different networks. Motivated by these benefits, a limited number of research studies have investigated the application of DTLs in DDoS attack detection for different types of networks [7, 8, 9]. As an initial work [10] in the related field, we proposed to utilize DTL based models to detect DDoS attacks on 5G network slicing environment.

This paper expands on several aspects of the preliminary work presented in [10]. First, we conduct comprehensive analysis of different DTL strategies by varying the number of layers that are retrained and by introducing different combination of new layers during the retraining process. Our analysis incorporates two distinct datasets encompassing both benign and malicious traffic flows specific to two different 5G networks. However, the two datasets that we utilized are highly skewed in the total number of samples; one having around six million samples whereas the other dataset has only a few hundred thousand samples. To ensure similar sizes for training/test sets in both datasets, we employed two different sampling methods focusing on consistency and reproducibility of our analysis in the first case and investigating the effects of variability and adaptability in the second case. Consequently, we iterate through both sample methods for each of our proposed DTL strategies to understand different ways of knowledge transfer and model enhancement. Finally, we expand our discussion of the related works by including recent developments on the application of DTL and detection of DDoS attacks in 5G and B5G networks.

The main contributions of this paper are as follows:

- We have generated a dataset comprising six million traffic flows (both attack and benign) from a 5G laboratory testbed [11], which is utilized as the source domain in developing our DTL models. The network within our testbed is bifurcated into two distinct slices, and traffic data is collated from both slices. On the other hand, the 5G-NIDD [12] dataset is used as the target domain that has a small amount

of annotated traffic data that is relevant to different DDoS attacks generated in a real 5G network.

- We explore various DL models, including Bidirectional Long Short-Term Memory (BiLSTM) [13], Convolutional Neural Network (CNN) [14], Residual Network (ResNet) [15], and Inception [16] as potential classifiers for DDoS attack detection. To do this, we use our dataset to create pre-trained models and DTL methods for different algorithms in the target dataset.

- We propose our methodology called DTL-5G based on two objectives. Objective 1 refers to consistency and reproducibility when using fixed samples and training/test sets. Objective 2 investigates the effects of variability and adaptation using dynamic resampling and changed training sets.

- We apply two different DTL strategies according to our objectives. Strategy 1 refers to fine-tuning without adding a new layer and only by freezing one or more layers and then retraining the model. Strategy 2 achieves fine-tuning by removing the last layer, adding one or more layers, and then retraining the model. Each strategy has three different scenarios by varying the number of layers that are retrained during fine-tuning.

- We perform a thorough evaluation to show the effectiveness of DL models in our generated dataset. Specifically, the BiLSTM model consistently outperforms other models and achieves an accuracy of 98.74%, a recall of 97.90%, a precision of 99.62%, and an F1-score of 98.75% for Objective 1. This indicates that the LSTM-based model is effective in detecting DDoS attacks in 5G networks. On the other hand, CNN-based models also show strong performance, particularly in terms of recall and F1-score for Objective 1, with the Inception achieving an accuracy of 97.23% and an F1-score of 97.27%. However, the CNN models required longer training times compared to LSTM based model.

- We evaluate both LSTM- and CNN-based models' performance on the 5G-NIDD dataset as a baseline for our DTL scenarios. For Objective 1, the Inception achieves the highest accuracy and F1-score, with values of 87.31% and 87.72%, respectively. This indicates that DL based models

perform well when sufficient amount training data is available like our generated dataset. However, the performance of the same DL based models deteriorates significantly when there is lack of sufficient amount training data such as the case of 5G-NIDD dataset.

- We conduct extensive evaluation to demonstrate the effectiveness of DTL strategies that enhance the performance of DL models on the 5G-NIDD dataset. In our evaluation, the DTL-based BiLSTM model demonstrates an enhancement of 13.90%, 21.48%, and 12.22% in accuracy, recall, and F1-score, respectively, compared to the baseline performance on the 5G-NIDD dataset. Furthermore, DTL-based Inception model demonstrates a 10.09% enhancement in precision compared to the same baseline.

The organization of this paper is structured as follows: Section 2 explores the current DTL-based techniques for 5G networks and also discusses existing research related to DDoS attacks on 5G networks. Section 3 introduces the fundamentals of DTL and the methodology of attack detection and classification based on DTL. Section 4 outlines the results and discussions. The paper is concluded with future direction in Section 5.

2. Related Work

This section discusses some previous works related to DTL and DDoS attack detection in the 5G network literature. As TL is a well-studied area of DL in other domains, such as computer vision and NLP, there are a multitude of research studies in the literature [17, 18]. However, little research has been done on applying TL, especially DTL, to DDoS detection problems in 5G networks. Hence, this paper studies the application of DTL in the context of IDS in 5G networks to detect DDoS attacks.

2.1. DTL Techniques on 5G-based Networks

The papers [19] and [20] both focus on improving IoT data classification and resource optimization in 5G networks, particularly under limited data conditions. The paper [19] introduces a method using TL with the AdaBoost algorithm to enhance classification accuracy, spectrum usage, and network throughput. Similarly, the paper [20] addresses network traffic classification challenges in 5G IoT systems by employing DTL with optimized pre-trained models like EfficientNet and Big Transfer (BiT) and achieving

near-complete accuracy with just 10% of labeled data. Experiments using the USTC-TFC2016 dataset further validate the effectiveness of this approach in data-restricted scenarios.

The study [21] explores the integration of IoT and 5G in the Industrial Internet of Things (IIoT), proposing TL to reduce the data and resources needed for training new models. TL is applied at two levels—machinery and networking— to enhance predictive maintenance and fault detection with pre-trained models on similar equipment and optimize the 5G infrastructure supporting IIoT. Similarly, the paper [22] examines network slicing in IIoT using a Deep Deterministic Policy Gradient (DDPG) algorithm enhanced by TL. The method optimizes slicing parameters like bandwidth and transmission power to improve QoS, energy efficiency, and reliability. TL accelerates learning across multiple gateways by leveraging knowledge from previously optimized contexts and ensuring faster convergence and better slice performance in dynamic environments.

The research [23, 24] proposes FortisEDoS, a framework to mitigate Economical Denial of Sustainability (EDoS) attacks in 5G network slicing. It combines a CG-GRU anomaly detection model with DTL to detect malicious behaviors and prevent unnecessary resource scaling. FortisEDoS includes a surveillance system for continuous monitoring, an auto-scaling module for real-time resource adjustment, and an EDoS Mitigator to validate scaling requests and block malicious ones. The framework effectively identifies complex EDoS attacks that mimic real traffic and reduce the risk of financial damage.

2.2. DDoS Attack Detection Techniques on 5G-based Networks

In a prior work [3], we evaluated the impact of DoS/DDoS attacks on 5G network slices, specifically on key performance indicators such as bandwidth and latency. Additionally, we created a new DoS attack detection system called SliceSecure, utilizing the LSTM model. The model demonstrated an impressive detection accuracy on the recently developed 5G dataset. We showcased the slice implementation using a 5G simulated testbed employing Free5GC [25] and UERANSIM [26]. This configuration enabled the researchers to both execute and quantify the immediate effects of DDoS attacks on network performance.

Several studies have proposed various models and frameworks to enhance DDoS detection in 5G and B5G networks. For instance, [2] employed a multilayer perceptron model with Pearson Correlation Coefficient (PCC) for

feature selection and achieved a high accuracy of 99.66% and a very low loss rate of 0.011 in classifying DDoS attacks in 5G and B5G networks. Similarly, [27] developed a detection method to protect infrastructure, tenants, and users across both the edge and core of 5G networks. The system, tested in real multi-tenant 5G infrastructure scenarios, demonstrated scalability, flexibility, and effectiveness in identifying and mitigating DDoS attacks.

Other research has focused on specific aspects of DDoS detection, such as traffic flow monitoring and anomaly detection. The paper [28] presented a method for detecting network devices involved in flood-based DDoS attacks by analyzing traffic flows at their origin and achieved high accuracy in distinguishing between normal operations and DDoS attacks, even with varying attack intensities. In a related study, [29] compared optimized individual ML models with ensemble learning models. Algorithms like Decision Trees, Random Forest, K-Nearest Neighbors, and Support Vector Machines were fine-tuned for accuracy. The results indicate that the optimized single ML model achieves a high detection accuracy and is at least 34 times faster than ensemble methods in detection time. The study [30] also proposed a framework for protecting MEC-based 5G networks from DDoS attacks by integrating anomaly detection with deep packet inspection. The orchestration technique ensures efficient resource allocation, allows the network to quickly adapt to security threats, and reduces the risk of compromising services for legitimate users.

DL-based approaches have also been explored extensively for DDoS detection. The paper [31] introduced DeepSecure, which uses an LSTM model to analyze traffic characteristics and predict normal or malicious behavior in a 5G network slicing environment. The method achieved a detection accuracy of 99.970% and a slice prediction accuracy of 98.798% and outperforms traditional approaches used in previous studies. The Secure5G model by [32] also utilized DL techniques to identify and mitigate DDoS attacks in a 5G network slicing environment. Secure5G effectively manages incoming connection requests, actively controls network slices, and pre-emptively isolates threats and redirects them to a quarantine slice to protect the 5G core network. Simulation results show a detection accuracy of over 98% using a limited dataset.

Other advanced techniques include the Modified Equilibrium Optimization Algorithm combined with DL for DDoS attack categorization in 5G networks, as presented in [33] and achieved a maximum accuracy of 97.60%. In addition, the works [34, 35] presented a method for detecting DDoS at-

tacks by managing network resources and isolating malicious traffic into a sinkhole-type slice. The DL models are trained in a 5G prototype built on the OpenAirInterface platform and the FlexRAN controller, which allows for dynamic control of RAN resources. The results show high detection accuracy and a significant reduction in throughput for malicious users, while maintaining high throughput for benign users.

Furthermore, Federated learning has been proposed as a novel approach to improve DDoS detection while preserving data privacy. The paper [36] introduced an unsupervised federated learning model that leverages the collective intelligence of devices to detect DDoS attacks in 5G core networks. The results showed improvements in detection rates in real-world scenarios.

In another work, the paper [37] proposed an autonomic security method for multi-tenant 5G networks, using a self-regulating control loop to manage DDoS attacks, even in scenarios with 256 attackers. The study [38] also introduced an inter-slice isolation mechanism to defend against DDoS attacks in the 5G core network. Tested through simulations and real experiments, the model effectively limits the impact of DDoS attacks, enhances network slice availability, and maintains network functionality. In addition, [39] addressed the issue of time-consuming authentication during inter-slice handovers, which can be exploited for DDoS attacks by analyzing UEs switching rates. Experimental results show that the method detects DDoS attacks with 91% accuracy and identifies compromised users with 96% accuracy.

Existing research in DTL, particularly within 5G and B5G networks, faces several significant limitations and gaps. One major challenge is the lack of a universal strategy for applying DTL across different domains, since its effectiveness is highly context-dependent and needs a trial-and-error approach, especially when determining how many layers to freeze and fine-tune. Additionally, while TL has been well-studied in fields like image and speech recognition, its application in emerging wireless technologies, such as 5G and B5G, remains relatively unexplored. Moreover, despite TL's potential to mitigate data scarcity, its success still hinges on the availability of suitable source domain data and reveals a need for methods that perform well in data-constrained environments. Finally, the scalability and computational complexity of DTL models pose challenges for real-time deployment in resource-limited settings like edge devices, which indicates a need for more efficient and lightweight models tailored to the constraints of 5G-based networks [6]. Addressing these gaps positions the current study as a critical contribution to advancing the field. This paper not only employs an ex-

tensive dataset and different DTL strategies, but it also specifically focuses on analyzing the enhancement of DL models in 5G-based networks. This study proposes valuable insights into the potential of these models to adapt to different network conditions.

3. Methodology

This section presents the DTL-5G methodology of the paper, fundamentals of DTL, data collection and pre-processing, base model construction, the proposed method, and DTL scenarios.

3.1. Fundamentals of DTL

Definition 1 (Task): Let us consider the feature space, denoted as χ, and the label space, labeled as γ. The dataset, represented as $X = \{x_1, \ldots, x_n\} \subset \chi$ and linked to labels $Y = \{y_1, \ldots, y_n\} \subset \gamma$. A task is a conditional distribution $P(Y \mid X)$ and defined by $T = \{Y, F(X)\}$, where F represents the learning objective predictive function. Within the framework of DTL, task refers to initial knowledge and also unknown knowledge in the learning process. DTL divides tasks into two categories: the source task represented as T_S, and the target task represented as T_T [40, 41].

Definition 2 (Domain): The marginal probability distribution over the dataset $X = \{x_1, \ldots, x_n\}$ is denoted as $P(X)$. The domain is represented as $D = \{X, P(X)\}$. In other words, the target domain, denoted as D_T, is defined as the set of elements X_T and $\mathrm{P}(X_T)$. Similarly, the source domain, denoted as D_S, is defined as the set of elements X_S and $\mathrm{P}(X_S)$. DTL revolves around two crucial domains: the source domain, represented as D_S, and the target domain, represented as D_T [40, 41].

Definition 3 (TL): TL refers to the process of using knowledge gained from a source domain D_S and a source task T_S to improve performance on a target domain D_T and a target task T_T. The primary objective of TL is to enhance the performance of the target task by leveraging the knowledge gained from the source domain and task. This is especially pertinent when the source and target domains $D_S \neq D_T$ or the source and target tasks or $T_S \neq T_T$ are dissimilar. If there is a disparity between D_S and D_T, it means that $X_S \neq X_T$ and/or $P(X_S) \neq \mathrm{P}(\mathrm{X}_T)$ [6]. This issue is referred to as heterogeneous TL. On the other hand, when $X_S = X_T$, but $P(X_S) \neq \mathrm{P}(\mathrm{X}_T)$, this situation is called homogeneous TL [42, 5]. When $T_S \neq T_T$, it indicates that the tasks are distinct. In addition, when $D_S \neq D_T$, the

Domain adaptation (DA) is the utilization of the source model's function F_S to enhance the target model's function F_T by incorporating knowledge from both D_S and D_T [40, 43].

Definition 4 (DTL): DTL aims to enhance the efficiency of the target function F_T by leveraging knowledge from F_S, which represents the source domain prediction function. It is especially important when the target domain D_T and the source domain D_S are different, or when the target task T_T differs from the source task T_S. In fact, when $D_S \neq D_T$ or $T_S \neq T_T$. The primary obstacle in DTL is to adjust to disparities between D_S and D_T or between T_S and T_T [40]. The formula for DTL, as described by [40], is given as follows:

$$D_S = \{X_S, P(X_S)\}, \quad T_S = \{Y_S, P(Y_S \mid X_S)\}$$
$$\rightarrow D_T = \{X_T, P(X_T)\}, \quad T_T = \{Y_T, P(Y_T \mid X_T)\}$$

3.1.1. TL Techniques

The main goal of TL is to apply acquired information from the source domain to enhance the learning process in the target domain. Three key factors must be considered to ensure optimal transfer performance, as follows [6]:

- **What to transfer:** This stage entails determining the specific components of the acquired knowledge from the source domain that should be transferred to the destination domain. Not all information derived from the source domain is advantageous for the target domain.
- **When to transfer:** It is crucial to ascertain the optimal moment for transferring knowledge. Transferring knowledge can be detrimental when the source and target domains are not closely related.
- **How to transfer:** The transfer procedure involves the actual transfer of identified information to the target domain.

The categorization of TL according to the factors mentioned leads to diverse methodologies [5, 6, 40, 42]:

- **Feature-based TL,** is a technique that entails generating a new feature representation by utilizing the existing features. There are two

methods to accomplish this: symmetrically, which involves transforming both the target and source features into a new representation, or asymmetrically, which involves transforming only the source features to align with the target features.

- **Parameter-based TL,** is an approach that focuses on transferring parameters, such as hyper-parameters or information, from the source model.

- **Relational-based TL,** refers to the process of transferring knowledge by comprehending the shared relationships between the source and target domains.

- **Instance-based TL,** refers to the process of modifying the weights of data samples from the source domain to reduce disparities between the source and target distributions.

As we mentioned earlier, "What to transfer" refers to the specific knowledge, such as features or parameters, that should be moved from the source to the target domain. This is closely linked to Feature-based TL, which transforms features to better align the source and target domains, and Parameter-based TL, which transfers model parameters that are beneficial to the target task. "When to transfer" involves determining the optimal timing for transferring knowledge and ensuring it only occurs when it's likely to be effective. This is addressed by Instance-based TL, which reweights data samples to align source and target distributions, and Relational-based TL, which transfers knowledge based on the similarity of relationships between domains. Finally, "How to transfer" focuses on the methodology used to implement the transfer. Feature-based TL and Parameter-based TL methods address this by either creating new feature representations or applying learned parameters directly to the target domain and optimizing the transfer process to enhance learning outcomes [5, 6, 40, 42].

In addition, TL can be categorized into three separate types according to the perspective of the problem [5, 6, 40, 42]:

- **Inductive TL,** refers to the process of utilizing knowledge from a source task to improve performance on a distinct target task.

- **Transductive TL,** is a situation in which the source and target tasks are identical, but they are applied in different domains.

- **Unsupervised TL,** is the method of acquiring knowledge from data without the use of labeled data in either the source or target domains.

3.1.2. DTL Techniques

The information stored in the structure and settings of a trained DNN can be applied to new tasks that are closely related to the field of DTL. Currently, there are three common strategies used in DTL, as follows [6]:

- **Utilizing ready-made pre-trained models:** It refers to models that have been trained on extensive datasets and may be used for related tasks without the need for additional training.
- **Utilizing pre-trained models as feature extractors:** In this context, pre-trained models are used to extract distinctive features from data, which are subsequently fed into a separate learning algorithm explicitly designed for the desired objective.
- **Fine-tuning pre-trained models:** This method entails using a pre-existing model and enhancing its performance by training it on the specific facts of the current task. Typically, the early layers of the pre-trained model remain unchanged, while only the latter levels are fine-tuned using the target data to prevent overfitting. Within this paper, we employed this particular technique.

3.2. Data Collection and Pre-processing

Source Dataset: Our research employs an extensive dataset [11, 3, 44, 10] created in a controlled 5G testbed environment, which serves as the source domain. The dataset comprises a diverse range of traffic types, encompassing both benign and malicious data, gathered from two network slices. The modeling of 5G network slicing scenarios utilized two opensource software, namely Free5GC and UERANSIM. In order to replicate benign internet traffic, we utilized an automated headless browser Python script to visit 500 various visited websites. This involved engaging in various online activities such as streaming movies, regular file copying and transfer activities, downloading and modifying data, conducting ICMP pings, SSH and other typical internet tasks. The dataset consists of around six million flow-based instances. In addition, the dataset contains 84 network traffic parameters and encompasses eight distinct types of DDoS attacks as follows: UDP flood, TCP syn, TCP push, TCP ack, TCP fin, TCP urg, TCP xmas, and TCP ymas. The data was

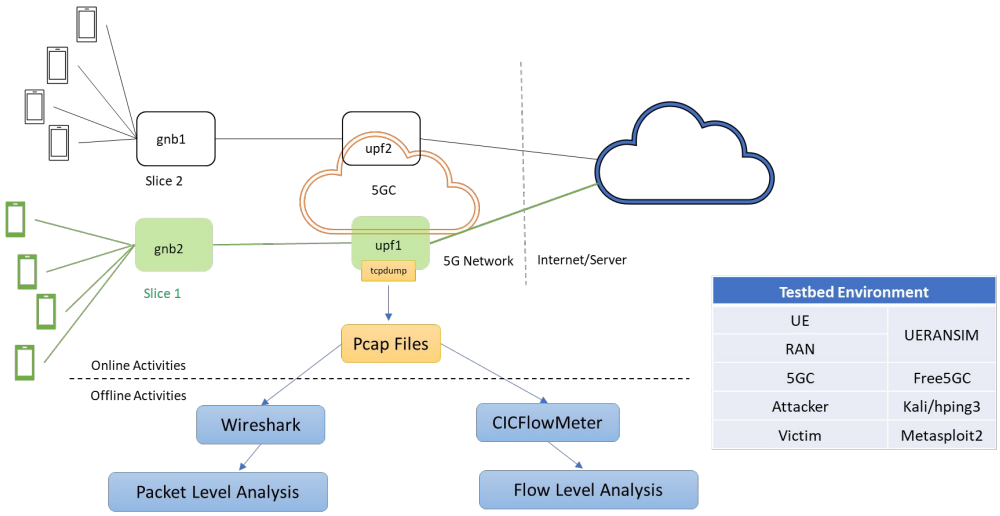

Figure 1: The Overall Architecture for the Methodology used on our dataset

collected with the hping3 [45] tool with precise parameters (i=120, d=350) and transformed into CSV format using the CICFlowMeter [46] traffic generator. These tools were selected because they are widely recognized and commonly used in DDoS attack research and provide a solid foundation for our experiments. Hping3, in particular, is favored for its flexibility and comprehensive documentation and makes it easy to configure for various types of network traffic generation, which is essential for simulating DDoS attacks. CICFlowMeter was chosen due to its effectiveness in extracting flow-based features, which are crucial for analyzing network traffic in DDoS scenarios. The parameters employed in our study are also well-established in the literature and ensure that our methodology aligns with standard practices in the field [3, 10, 12, 47, 48]. This careful selection of tools and parameters enhances the reliability and replicability of our results. Our dataset undergoes a meticulous conversion process from packet-level captures to flow-level forms for data processing. The overall architecture for the methodology used in our dataset is shown in Figure 1.

Target Dataset: The 5G-NIDD [12], or 5G Network Intrusion Detection Dataset, serves as our target domain. The dataset offered is meticulously cu-

Table 1: Number of instances in each Source and Target dataset

Our Dataset		5G-NIDD Dataset	
Attack Type	Flow Instances	Attack Type	Flow Instances
Benign	967567	Benign	15212
UDP flood	381073	UDP Flood	194946
TCP YMas	524288	SYN Flood	7566
TCP XMas	524288	GoldenEye	72499
TCP Urg	470757	ICMP flood	2
TCP SYN	405774	Slowloris	8669
TCP ACK	499729	UDP scan	33
TCP Push	524288	SYN scan	75
TCP FIN	916050	TCP Connect	189
		TorShammer	31686

rated and originates from a genuine 5G testing setting. This dataset offers comprehensive insights into the intricate patterns exhibited by both benign and malicious data traffic within a fully functional 5G network. The benign traffic, comprising protocols such as HTTP, HTTPS, SSH, and SFTP, is generated using authentic mobile devices to ensure the authenticity and reliability of the data. The dataset encompasses many attack scenarios, encompassing multiple types of DoS attacks and port scans. The types of DoS attacks include ICMP flood, UDP flood, SYN flood, HTTP flood, and Slowrate DoS. The authors conduct three different forms of scan attack during the process of data collection. The types of attacks being used are SYN scan, TCP connect scan, and UDP scan. This comprehensive technique enhances the dataset's pertinence to current research and its appropriateness for developing efficient AI/ML-based IDSs. The 5G-NIDD dataset, like our dataset, undergoes a rigorous conversion process where packet-level captures are transformed into flow-level forms to facilitate data processing. The 5G-NIDD protocol encompasses attacks at both the application layer and the transport layer, but our dataset exclusively includes attacks at the transport layer. The number of flow samples in 5G-NIDD is much smaller than those in our dataset, making 5G-NIDD suitable for the target dataset in DTL scenarios. Table I displays the quantity of benign and attack traffic occurrences in both our dataset and the 5G-NIDD dataset.

3.2.1. Maximum Mean Discrepancy (MMD)

The Maximum Mean Discrepancy (MMD) is a metric used to quantify the difference between two distributions [49]. We calculate the MMD score, which compares the distribution of the source and target datasets. The distributions are identical when the MMD score is 0, and distinct when the MMD number exceeds 0. In our experiments, we compute the MMD score for the two datasets that we have chosen. The MMD score of 0.2605 indicates the level of dissimilarity between the two distributions in both datasets.

3.2.2. Feature Extraction and Optimization

In this section, CICFlowMeter was employed to carry out an initial processing step on the dataset and convert the collected .pcap files to the .csv format, in which 84 distinct traffic features could be identified. These characteristics were vital for detecting potential DDoS attacks. As was the case with other research studies, the research involved the identification and selection of important features based on their relation to patterns of DDoS attacks. Improving the feature set resulted in more precise predictions by the model, which led to feature optimization. Significance in detecting DDoS attacks was the criterion for the initial selection of eight features from the original collection. Metadata aspects were removed to prioritize intrinsic data attributes with a higher predictive value for network threats. Principal Component Analysis (PCA) helped identify the most important features, including flow duration, forward packet length, standard deviation, etc. The PCA highlighted their variability and impact on the model's ability to distinguish between normal and DDoS attack traffic. Finally, eight features were selected in total, which are presented in Table 2 [10, 44].

3.2.3. Data Cleaning

This section aims to remove duplicate and unnecessary features. For this purpose, we removed abnormalities such as infinite values and modified them to "NaNs" to ensure clean data. Ensuring the quality and relevance of the data inputted into the model is crucial to optimizing the accuracy and efficiency of subsequent phases.

3.2.4. Data Resizing and Resampling

Data resizing is the process of changing the dimensions of the input data to match a specific shape required for the model. In this paper, since our

Table 2: Network Features Description

Serial	Feature	Description
1	Flow duration	Duration of the flow in Microseconds
2	Total Fwd Packet	Total number of packets in the forward direction
3	Total Bwd packets	Total number of packets in the backward direction
4	Total Length of Fwd Packet	Total size of packet in forward direction
5	Total Length of Bwd Packet	Total size of packet in backward direction
6	ACK Flag Cnt	Number of packets with ACK flag
7	Fwd Packet Length Std	Standard deviation of packet length in the forward direction
8	Protocol	Network protocol used

network traffic shape is one-dimensional (1D), we resized the data shape for DL models to 1D.

In addition, to address the imbalance between benign and attack traffic samples, the data for each attack type is downsampled to a lower size that is proportional to the benign traffic. This helps mitigate model bias towards the class, which occurs more frequently. This process resamples each category of attack traffic samples to reduce the number of samples, aiming for a dataset that is approximately one-eighth the size of the benign samples in order to generate a more balanced dataset. For this purpose, given the unique characteristics of the target domain, which is the 5G-NIDD dataset in this instance, the size of the test set in the source domain was reduced by downsampling. This adaptation helps to make the test set size more closely match the size of the target domain. Using random shuffling and selection of indices guarantees that the test subset is both representative and truly random.

3.2.5. Data Rescaling

Data rescaling is the process of adjusting the scale of the input data so that it fits within a predetermined range. It is very important to prevent any bias toward specific features during training the model. Methods such as

min-max scaling, standardization, or normalization can achieve this. In this paper, the process of normalization and standardization is carried out using the "StandardScaler" module from the sklearn library. This module standardizes the features by removing the mean of the training samples (mean=0) and scaling them to have a standard deviation of one (standard_deviation = 1). This guarantees that the model does not develop a bias towards characteristics that have greater scales. The same transformation that is employed in the training set is also applied to the testing set. This ensures that both datasets are standardized in a uniform manner and ensures consistent input for the training and evaluation of the model.

3.3. Test and Train Splitting

The data is partitioned between training and testing sets to ensure that the model is evaluated on unfamiliar data and emulates real-world performance. To achieve this, 20% of the data is allocated for testing purposes, and 80% of the data is allocated for the training part. The "stratify" option is also utilized to guarantee that the train and test sets preserve the same class distribution as the original dataset. In addition, it preserves a balanced distribution of classes across both source and target datasets.

3.4. Proposed Method

The study is organized based on two main objectives, each intended to investigate distinct facets of model evaluation.

Objective 1: The purpose of this objective is to ensure the dependability of the evaluation process by implementing consistent sampling approaches, and establishing preset sizes for the training and test sets. The research begins by collecting an equal number of samples for each type of cyber-attack, as well as for benign cases, in order to achieve an equal sample distribution. This methodology guarantees a dataset that is evenly distributed, which is important for eradicating discrimination towards any one category during the process of training and evaluating the model. In order to keep the dependability and consistency of the experiments, a consistent sampling methodology used in each iteration. In this paper, we performed five iterations of model assessment, using a consistent "random_state" for each iteration. The "random_state" values for each iteration were set at 42, 142, 12, 4, and 80, respectively. The same settings are used for each model. Following this step ensures that the outcomes can be replicated and any differences in performance can be ascribed to the model's actions rather than changes in the

selection of samples. The proportions of the training and test sets were predetermined and consistent for both datasets used in the study. It is crucial to define fixed sizes for these sets in order to ensure that the evaluation of model performance is consistent and can be compared across all experiments.

Objective 2: In contrast, Objective 2 acknowledges and incorporates the natural variability and unpredictability of cyber-attacks by incorporating dynamic components into the experimental settings. This encompasses the utilization of variable resampling techniques, the utilization of unfixed training datasets, and the examination of models' performance as the input data changes. The main goal of objective 2 is to replicate real-life situations where IDS need to adjust to new and emerging threats constantly. This allows us to evaluate the resilience and flexibility of the models. Just like Objective 1, this approach requires gathering an equal number of samples from all types of attacks and benign cases. Objective 1 uses a fixed sampling method, while Objective 2 utilizes dynamic resampling for each cycle of model testing. In this study, we performed five iterations, the same as Objective 1, with each iteration not predefining the "random_state". The objective of this technique is to assess the performance of the models in different settings in order to evaluate their resilience and consistency in uncertain scenarios. As a result, the training sets was not constant during different runs, unlike the previous objective. This method enables examining how the differences in the training part impact the effectiveness of the model. This is done to see how the model's capacity can be adapted to new data.

In Objective 2, similar to Objective 1, the size of the test set was modified to meet the specific size of the 5G-NIDD dataset. The main difference between Objectives 1 and 2 is based on the methodology used for sample selection and the management of the test and train parts, which is represented in Table 3. Objective 1 prioritizes consistency and reproducibility by using fixed samples and training/test sets. On the other hand, Objective 2 investigates the effects of variability and adaptation by using dynamic resampling and unfixed training sets. Both Objectives, meanwhile, align with the principles of fair sample allocation, adaption to the desired domain, and the strategic use of DTL.

3.5. Base Model Construction

The BiLSTM and CNN-based models, including CNN, ResNet, and Inception, were implemented and employed for one-dimensional (1D) data to

Table 3: Techniques Comparison for Objective 1 and Objective 2

Objective 1	**Objective 2**
✓ Take equal samples for all types of attacks and benign	✓ Take equal samples for all types of attacks and benign
✓ Fixed sampling in each iteration for same model (5 iterations with random_state = 42, 142, 12, 4, 80)	✓ Resampling in each iteration (5 iterations without using random_state)
✓ Fixed training sets and test sets size from both datasets	✓ Keep test sets unchanged as used for objective 1
✓ Consider test set sizes with down sampling based on target domain (5G-NIDD) test set size	✓ Consider test set sizes with down sampling based on target domain (5G-NIDD) test set size
✓ Use the best model created from our dataset for DTL section	✓ Use the best model created from our dataset for DTL section

construct base models utilizing our dataset as the source domain. We followed [50, 51] to implement architectures as shown in Table 4. The selection of BiLSTM, CNN, ResNet, and Inception models in this paper is grounded in their proven effectiveness in similar studies and their prior success in related applications [3, 6, 10, 15, 16, 52, 53, 54, 55]. These models are particularly suitable for the problem at hand due to their distinct architectures, which have been widely adopted in related works. Specifically, the two types of layers we focus on in this paper—LSTM-based and CNN-based—are among the most popular and extensively used in the field of cybersecurity and network threat detection. LSTM and CNN both layers have different ways to generate features. Our work investigates which type of feature generation works best in our datasets. We investigated these structures because we were interested in seeing how DNNs perform on the flow-based dataset. In addition, starting with multiple models provides different capabilities and benchmarks their performance to find the best approach for our dataset. The model's performance will improve if the domains in which DTL may be applied are more similar, as this will address the issue of insufficient data examples and save training time.

During this period, we encountered various sequential steps: 1) We started training on the source dataset without using any pre-existing model and stored it as a base model. The mentioned models are employed in the DTL

Table 4: Details for all Pre-trained Models

Model	Layers	Total Parameters	Trainable	Non-Trainable
CNN	9	6562	6562	0
ResNet	45	506818	504258	2560
Inception	73	427650	425602	2048
BiLSTM	9	99970	99970	0

part of the target dataset. 2) Additionally, we conducted training on the target dataset using the same models, starting from the beginning, in order to assess the improvement between the IDS based on DTL strategy and the same approach without DTL.

3.6. DTL Strategies and Scenarios

The source and target domains are the same (benign and attack) but involve different flows. This is why, in Strategy 1, we opted to freeze most layers while re-training only a few. By doing so, we can leverage the knowledge learned from the source dataset when applied to the target dataset. Freezing the majority of the model's layers ensures that we primarily utilize the information acquired from the source dataset. However, as noted in the literature [5, 6, 42, 52, 56, 57, 58, 59], there is no definitive method to determine in advance how many layers should be frozen and how many should be fine-tuned for effective DTL. This uncertainty led us to explore various scenarios through additional experiments.

Additionally, we are interested in investigating whether increasing the number of layers in the pre-trained source model can enhance its performance in the target domain. Therefore, in Strategy 2, we removed the last layer of the source model and added new layers. This approach allows our DTL strategy to be as comprehensive as possible since we could not predict which scenario would be most applicable and prompts us to consider multiple possible scenarios.

Strategy 1: Freezing one or more layers and retraining the model (fine-tuning without adding new layers)

This method involves the selective freezing of specific layers in pre-trained models in order to preserve the features they have learned. We retrain the remaining layers using the weights generated in the pre-train stage after freezing the layers. This strategy capitalizes on the strength and reliability of pre-trained features, while also enabling customization and fine-tuning to

better align with the specific requirements of the new work. According to strategy 1, we have three DTL-based scenarios as follows:

- **DTL0: Freeze All Layers Except the Last One and Retrain**

  In this scenario, we fine-tuned the model's final decision-making layer. This process entails solely modifying the weights of the last layer while leaving the remaining parameters of the model unaltered. This scenario enables the model to adjust its final output to the task at hand and retain the acquired features from the pre-trained models. The initial layers of the model, which are responsible for extracting features from the input data, are kept fixed and not modified. This is advantageous when the characteristics obtained from the initial training task are applicable to the new task.

- **DTL1: Freeze All Layers Except the Last 33% Layers and Retrain**

  In this scenario, all layers for all pre-trained models are frozen except the last 33% of layers, and then unfrozen layers are retrained. This technique allows the model to adjust its deeper layers to the new task and retain the initial feature extraction layers.

- **DTL2: Freeze All Layers Except the Last 66% Layers and Retrain**

  This scenario is very similar to DTL1. However, the percentage of layers that go through the retraining process is now increased to 66%. This scenario allows more flexibility in adapting the model to the new task compared to DTL1, as a larger portion of the model's parameters are updated.

The percentages of 33% and 66% for freezing layers were empirically selected based on our previous research [10] using the ResNet architecture. We aimed to explore how different levels of layer freezing would impact the performance of the model. The ResNet architecture served as a foundation for determining these percentages, which we then proportionally applied to other models in our study. This approach allowed us to systematically investigate the effects of varying the extent of layer freezing on model performance across different architectures. Our decision to focus on these specific scenarios was

informed by the need to balance the retention of learned features with the adaptability of the model to the target domain.

Strategy 2: Freezing Layers, removing the last layer, adding new layers, and retraining the model (fine-tuning with adding new layers)

In strategy 2 based on DTL, we froze all layers except the last one and removed the last layer of the pre-trained models. Upon removing this layer, we continue adding new layer(s) that have been specifically customized for the new task. These additional layers can enhance the model's ability and adaptability to obtain new patterns that are relevant to the new task. After that, the modified model undergoes a process of retraining. It guarantees integration between the newly added layers and the existing layers and ensures that the model efficiently adapts to the new task demands. Similar to strategy 1, we have three DTL-based scenarios for strategy 2, as follows:

- **DTL3: Freeze Layers and Remove the Last Layer, Add One New Layer, and Retrain**

  In this scenario, we froze all layers except the last one and then replaced the last layer with a new layer whose weights are initialized randomly. The new layer is finally trained to adapt the model to the new task. This can help in situations where the output structure of the new task differs from the original task.

- **DTL4: Freeze Layers and Remove Last Layer, Add Two New Layers, and Retrain**

  This approach is similar to DTL3, but two new layers are added instead of adding one new layer. In this scenario, we froze all layers except the last one and replaced it with two new layers whose weights are initialized randomly. This increases the model's capacity to learn complex patterns specific to the new task and provides more flexibility in the adaptation process.

- **DTL5: Freeze Layers and Remove Last Layer, Add Three New Layers, and Retrain**

  Extending the concept from DTL3 and DTL4, three new layers are added after removing the last layer. This scenario offers the highest flexibility and capacity for learning new patterns, making it suitable for tasks that require significant changes in the model's output behavior.

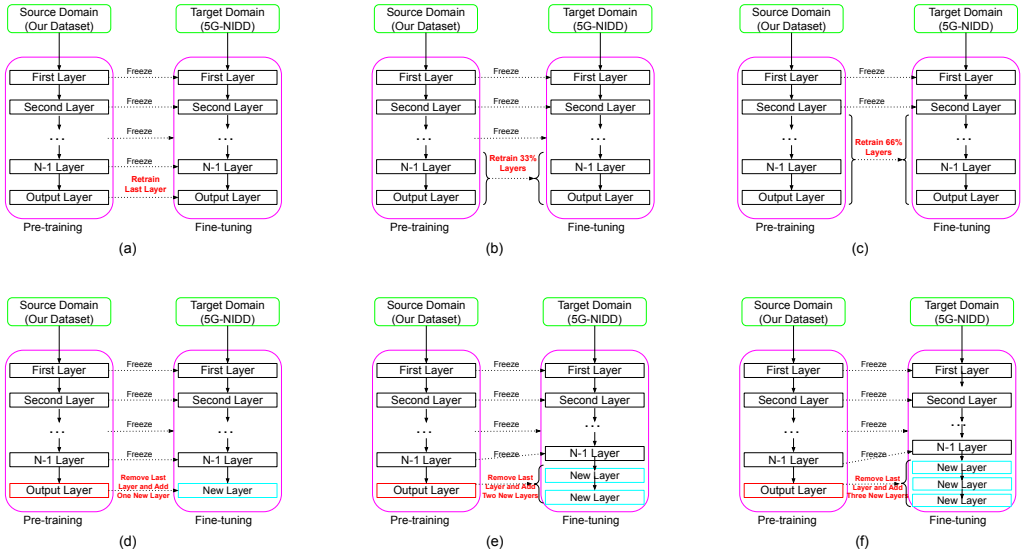

Figure 2: The overall process for all scenarios a) DTL0 b) DTL1 c) DTL2 d) DTL3 e) DTL4 f) DTL5

The overall process for all DTL-based scenarios are presented in Figure 2.

3.7. Limitations and Challenges

One of the primary challenges we face is determining the optimal number of layers to freeze and the appropriate number of layers to add for fine-tuning. This uncertainty presents a significant limitation in our approach, particularly due to the increased computational demands associated with retraining more layers and adding new ones. Additionally, selecting initial weights and performing hyperparameter tuning further intensifies the computational load, given the inherently demanding nature of training.

Another potential limitation of our strategies is their effectiveness across different scenarios. Specifically, if the data distributions between the source and target datasets are significantly different, our methodology may not perform as expected. Our current approach assumes a certain similarity in data distributions, as indicated by the MMD score, which may not hold true in other contexts. Consequently, while our strategies may be effective with datasets that have similar distributions, they may yield suboptimal results with datasets that differ significantly. Despite these potential limitations, conducting experiments with different datasets can provide valuable insights

into the effectiveness of our strategies across various contexts. Such experiments would help identify which strategies might be more suitable for different types of data, ultimately informing the development of more robust approaches in future research.

4. Results and Discussion

The BiLSTM and CNN-based algorithms, including CNN, ResNet, and Inception algorithms were developed using Keras, which is a Python library running on TensorFlow. The models were trained on Google Collaboratory cloud servers and Compute Canada [60], as well as on a laptop equipped with an 11th Generation Intel Core i7 processor and 16GB RAM. Both source and target datasets contain 8 features (same for both) in addition to the label, which contribute to the modeling process. The labels 'attack' and 'benign' are represented by binary numbers. For training and validation purposes, a test split of 20% of data and 80% training data, as well as a validation split of 0.2, are utilized.

Furthermore, we focused on optimizing key hyperparameters through a trial-and-error approach to achieve the highest performance. Among the various hyperparameters tested, the learning rate (set to 1e-5), the number of convolutional layers, the L2 regularization strength (set to 0.001), and the dropout rate (set to 0.5) were found to have the most substantial effect on the model's performance. Adam optimizer was used with a learning rate of 1e-5, which played a critical role in controlling the convergence of the model during training, while the number of convolutional layers determined the model's ability to capture complex features. The L2 regularization and dropout were essential for preventing overfitting and contributing to the model's generalization ability. These hyperparameters were carefully selected to balance performance and stability to ensure that the model achieved optimal results. In addition, the training method is executed for 200 epochs, incorporating early stopping mechanisms for augmentation. The "EarlyStopping" callback is employed to stop training if the validation loss fails to demonstrate improvement, thereby preventing overfitting and conserving computational resources.

In our experiments, we conducted an empirical evaluation using both our dataset and the 5G-NIDD dataset to assess the performance of each pre-trained model. For this purpose, we used well-known metrics, namely accuracy, recall, precision, and the F1-score, to evaluate the performance of

our proposed method. Specifically, recall and F1-score are critical metrics for evaluating DDoS detection in 5G-based networks due to their direct impact on the network's QoS. High recall is essential because False Negatives (FN)—instances where the model fails to identify an attack—can result in undetected threats that may have detrimental effects on the network. Similarly, a strong F1-score balances recall and precision and reduces the likelihood of False Positives (FP). False positives, where benign traffic is misclassified as malicious, can lead to unnecessary mitigation actions that negatively impact QoS. Therefore, these metrics are particularly important in ensuring the robustness and reliability of DDoS detection strategies in 5G environments, justifying their use in our evaluation strategy. Table 5 summarizes the evaluation metrics utilized in this paper. Furthermore, the total time of training (in minutes) and inference time (in milliseconds) are also calculated. The total inference time on both the central processing unit (CPU) and graphics processing unit (GPU) are recorded and documented.

Table 5: Definitions of Evaluation Metrics

Metric	**Formula**	**Definition**
True Positive (TP)	-	number of actual attack samples that are correctly detected as attacks
True Negative (TN)	-	number of normal samples that are correctly detected as normal
False Positive (FP)	-	number of normal samples that are incorrectly detected as attacks
False Negative (FN)	-	number of actual attack samples that are incorrectly detected as normal
Accuracy	$\frac{TP+TN}{TN+FN+TP+FP}$	percentage of accurate predictions among all predictions
Precision	$\frac{TP}{TP+FP}$	percentage of accurate predictions among all positive predictions
Recall	$\frac{TP}{TP+FN}$	percentage of positive labels that the classifier correctly predicted to be positive
F1-score	$\frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$	harmonic mean of precision and recall.

We analyzed the improvements across different metrics in an evaluation

of model performance in different base models and scenarios from DTL0 to DTL5 on the 5G-NIDD dataset to determine which models significantly improved performance. The simulation results are presented in Tables 6 to 13. Among the models considered—CNN, ResNet, Inception, and BiLSTM—distinct improvements (highlighted) were evident in accuracy, recall, precision, and F1-Score.

Table 6: Performance of Different Models on Our Dataset

Model	Objective	Our dataset						
		Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Train Time(m)	Inf CPU(ms)	Inf GPU(ms)
CNN	1	94.10	91.56	97.10	94.25	228.04	0.11	0.07
	2	94.35	92.02	97.16	94.52	260.04	0.12	0.07
ResNet	1	95.67	92.76	99.16	95.84	807.65	0.29	0.12
	2	97.07	95.07	99.36	97.16	769.76	0.28	0.11
Inception	1	97.23	96.47	98.08	97.27	914.44	0.27	0.16
	2	96.88	96.24	97.54	96.88	994.77	0.28	0.16
BiLSTM	1	98.74	97.90	99.62	98.75	466.85	0.22	0.10
	2	98.45	97.46	99.48	98.46	518.79	0.19	0.10

Table 7: Performance of Different Models on the 5G-NIDD Dataset

Model	Objective	5G-NIDD dataset						
		Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Train Time(m)	Inf CPU(ms)	Inf GPU(ms)
CNN	1	85.31	86.43	84.18	85.26	7.80	0.09	0.07
	2	85.13	86.59	83.57	85.02	7.37	0.09	0.07
ResNet	1	85.85	93.19	78.58	84.80	6.80	0.21	0.11
	2	81.97	88.49	76.62	80.95	6.41	0.21	0.10
Inception	1	87.31	85.86	89.85	87.72	8.44	0.24	0.14
	2	85.88	88.54	83.59	85.57	7.33	0.23	0.13
BiLSTM	1	84.00	79.59	90.55	85.33	11.67	0.15	0.09
	2	83.29	79.81	89.75	84.44	11.65	0.15	0.09

Table 8: Performance of Different Models on the 5G-NIDD Dataset with DTL0

Model	Objective	5G-NIDD dataset with TL0						
		Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Train Time(m)	Inf CPU(ms)	Inf GPU(ms)
CNN	1	**86.91**	84.70	**90.10**	**87.32**	7.95	0.09	**0.06**
	2	**87.08**	85.25	**89.21**	**87.40**	8.04	0.09	**0.06**
ResNet	1	**85.98**	93.13	77.71	84.72	8.16	0.22	0.10
	2	**84.76**	**92.76**	75.41	**83.11**	7.72	0.24	0.11
Inception	1	**93.56**	**98.93**	88.07	**93.18**	26.02	0.24	0.14
	2	**93.90**	**98.37**	**89.29**	**93.61**	29.63	0.23	0.14
BiLSTM	1	**94.87**	**96.50**	**94.98**	**94.78**	12.94	0.15	0.08
	2	**94.68**	**96.85**	**92.37**	**94.55**	13.84	0.16	0.09

Table 9: Performance of Different Models on the 5G-NIDD Dataset with DTL1

Model	Objective	5G-NIDD dataset with TL1						
		Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Train Time(m)	Inf CPU(ms)	Inf GPU(ms)
CNN	1	**86.92**	84.67	**90.18**	**87.33**	8.21	0.09	0.07
	2	**87.32**	85.71	**89.55**	**87.59**	8.21	**0.08**	**0.06**
ResNet	1	**87.00**	**93.98**	79.13	**85.84**	8.32	0.22	**0.10**
	2	**85.12**	**92.59**	76.35	**83.69**	6.34	0.22	0.11
Inception	1	**94.73**	**98.89**	**90.47**	**94.49**	24.26	0.24	0.14
	2	**93.81**	**98.42**	**89.05**	**93.49**	24.41	0.24	0.14
BiLSTM	1	**95.04**	**96.60**	**93.38**	**94.96**	13.67	0.16	0.09
	2	**94.87**	**96.66**	**92.94**	**94.76**	13.32	0.16	0.09

Table 10: Performance of Different Models on the 5G-NIDD Dataset with DTL2

Model	Objective	5G-NIDD dataset with TL2						
		Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Train Time(m)	Inf CPU(ms)	Inf GPU(ms)
CNN	1	**86.86**	84.72	**89.93**	**87.25**	7.82	**0.08**	0.07
	2	**87.21**	85.58	**89.51**	**87.50**	7.83	**0.08**	**0.06**
ResNet	1	**86.12**	**94.31**	76.87	84.70	9.06	0.22	**0.10**
	2	**84.94**	**92.85**	75.71	**83.40**	6.24	0.23	0.11
Inception	1	**94.70**	**98.84**	**90.46**	**94.47**	24.96	0.24	0.14
	2	**93.20**	**98.67**	**87.58**	**92.78**	22.12	0.23	0.14
BiLSTM	1	**94.99**	**96.69**	**93.17**	**94.90**	12.97	0.15	**0.08**
	2	**94.60**	**96.72**	**92.33**	**94.47**	12.70	0.15	0.09

Table 11: Performance of Different Models on the 5G-NIDD Dataset with DTL3

Model	Objective	5G-NIDD dataset with TL3						
		Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Train Time(m)	Inf CPU(ms)	Inf GPU(ms)
CNN	1	85.07	83.72	**87.02**	**85.28**	7.99	**0.08**	0.07
	2	84.06	83.69	**84.55**	84.08	7.79	**0.08**	**0.06**
ResNet	1	**86.32**	**93.75**	77.87	**85.04**	7.01	0.22	**0.10**
	2	**84.94**	**92.04**	76.54	**83.54**	7.08	0.23	0.11
Inception	1	**94.67**	**97.67**	**91.53**	**94.49**	24.33	**0.23**	**0.13**
	2	**94.31**	**96.44**	**92.02**	**94.18**	24.61	0.24	0.14
BiLSTM	1	**93.58**	**92.36**	**95.18**	**93.71**	13.03	0.15	0.09
	2	**94.21**	**94.64**	**93.74**	**94.19**	12.92	0.15	0.09

Based on Tables 6 to 13, we can identify which models excel in each metric by examining each model's performance improvements.

In Table 6, the BiLSTM model achieved superior performance on our dataset with an accuracy of 98.74%, recall of 97.90%, precision of 99.62%, and an F1-score of 98.75% for Objective 1. It also had acceptable training times of 466.85 minutes and low inference times (0.22 ms on the CPU and 0.10 ms on the GPU). In addition, Inception had good performance, especially for Objective 1, with an accuracy of 97.23% and an F1-score of 97.27%, although

Table 12: Performance of Different Models on the 5G-NIDD Dataset with DTL4

Model	Objective	5G-NIDD dataset with TL4						
		Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Train Time(m)	Inf CPU(ms)	Inf GPU(ms)
CNN	1	**86.49**	85.75	**87.52**	**86.62**	8.48	0.09	0.07
	2	**86.59**	85.70	**87.84**	**86.75**	8.17	**0.08**	**0.06**
ResNet	1	**86.51**	**93.74**	78.26	**85.28**	**6.34**	0.22	0.11
	2	**85.42**	**93.20**	76.42	**83.97**	**5.15**	0.22	0.10
Inception	1	**94.45**	**97.89**	**90.88**	**94.24**	27.20	0.24	0.14
	2	**94.18**	**96.43**	**91.76**	**94.04**	20.63	0.23	0.13
BiLSTM	1	**92.21**	**91.34**	**93.54**	**92.33**	13.48	0.16	0.09
	2	**94.46**	**95.31**	**93.52**	**94.40**	13.10	0.15	**0.08**

Table 13: Performance of Different Models on the 5G-NIDD Dataset with DTL5

Model	Objective	5G-NIDD dataset with TL5						
		Accuracy(%)	Recall(%)	Precision(%)	F1-score(%)	Train Time(m)	Inf CPU(ms)	Inf GPU(ms)
CNN	1	**87.53**	**92.47**	81.76	**86.77**	5.59	0.09	0.07
	2	**87.54**	**91.11**	83.40	**86.99**	8.07	0.09	0.07
ResNet	1	**85.93**	**93.54**	77.21	84.59	**6.53**	0.23	0.11
	2	**84.94**	**90.80**	**77.76**	**83.77**	**5.89**	0.23	0.11
Inception	1	**94.37**	**97.61**	**90.99**	**94.18**	20.59	**0.23**	0.14
	2	**93.83**	**95.86**	**91.64**	**93.69**	23.71	0.24	0.14
BiLSTM	1	**93.04**	**91.24**	**95.45**	**93.24**	13.71	0.15	0.09
	2	**93.97**	**95.78**	**92.00**	**93.85**	13.47	0.16	0.09

it had the longest training time. The ResNet and CNN models showed good performances but were slightly lower compared to the BiLSTM and Inception models.

In our study, the baseline is defined as the performance of the DL models on 5G-NIDD dataset without applying any DTL techniques. We evaluated the performance of various models against this baseline. Table 7 displays each model's best performance on the 5G-NIDD dataset. In this paper, 5G-NIDD is considered as a baseline and all . For Objective 1, Inception achieved the highest accuracy and F1-score, with 87.31% and 87.72%, respectively. In addition, ResNet had the best performance for Objective 1, with a high recall of 93.19%. BiLSTM also achieved its best performance in terms of precision, at 90.55% for Objective 1. Inference times for all models were low, with minor differences between CPU and GPU times, and training times varied, with the longest training time for the BiLSTM model.

Accuracy: According to the accuracy, the BiLSTM model showed the most significant improvement over baseline. Specifically, under scenario DTL1 Objective 2, BiLSTM improved its accuracy over baseline from 83.29%

to 94.87%, with an increase of 13.90%. The results demonstrate its robust adaptation to the 5G-NIDD dataset through the DTL strategies and scenarios.

Recall: Based on recall, the BiLSTM again indicates significant improvements over baseline, particularly under scenario DTL2 Objective 1, where it improved over baseline from 79.59% to 96.69%, an increase of 21.48%. The big jump in recall for BiLSTM shows that it can pick out more relevant samples from the dataset, which is important for tasks that need to be sensitive to certain conditions or features.

Precision: In terms of precision, the Inception model demonstrates the best improvement over baseline under DTL3 Objective 2, increasing from 83.59% to 92.02%, an increase of 10.09%. Inception's notable improvement in precision reflects its ability to minimize false positives and maintain high accuracy in its predictions.

F1-Score: In the case of the F1-Score, the BiLSTM model again has improvements over baseline, especially under DTL1 Objective 2, where it improved from 84.44% to 94.76%, an increase of 12.22%. BiLSTM's improvement over baseline in F1-score signifies that it is suitable for diverse applications that require both high precision and recall. To conclude, the analysis indicates the strengths of each model across different performance evaluation metrics. It is obvious that BiLSTM generally provides the most robust improvements over baseline, especially in Accuracy, recall, and F1-score, while Inception leads in precision. This distinction helps us to select the appropriate model and DTL strategy based on the specific needs of the task at hand.

Train Time: CNN and Inception under the DTL0 scenario have minimal increases in training time over baseline. CNN consistently has the shortest training times across all DTL strategies, and it is efficient in terms of training time. On the other hand, Inception consistently has the longest training times across all DTL strategies, which reflects the complexity and longer training time needed for data processing. However, ResNet has the lowest training times in DTL4 and DTL5 scenarios based on both objectives 1 and 2.

Inference Time: CNN has some improvements over baseline in terms of CPU and GPU, and it offers the shortest inference times under all DTL strategies and is suitable for applications that demand fast predictions. In addition, ResNet and Inception have the longest Inference times across all DTL strategies on both CPU and GPU.

In this paper, we employed both BiLSTM and CNN-based models, including variants such as ResNet and Inception, to evaluate their performance on datasets. Through extensive experimentation, it is evident that the BiLSTM model consistently outperforms the CNN-based models across various metrics. This superior performance can be attributed to the unique capabilities of the BiLSTM architecture. BiLSTMs can generate a more nuanced feature representation compared to 1D CNN by processing the data in both forward and backward directions. This bidirectional processing can capture more complex patterns in the data. This could be one of the reasons why BiLSTM is performing better than CNN-based models. Furthermore, BiLSTM has a gating mechanism where three type of gates are used. One of the gates is called "forget gate" that helps the model to forget irrelevant features. Therefore, BiLSTM has this built-in ability to select relevant features which may give BiLSTM extra advantage compared to the CNN-based models.

Table 14 summarizes the overall improvements over baseline for all DTL scenarios in this paper. Based on the analysis, Strategy 1 and the DTL1 scenario emerge as the most effective combination to improve model performance on the 5G-NIDD dataset. In particular, DTL1 in Strategy 1, which freezes all but the last 33% of layers and then retrains, gets the best results in key metrics, especially the F1-score, which went up by 12.22% for the BiLSTM model. This approach effectively balances preserving the learned features from the source domain while allowing sufficient flexibility in the model to adapt to the specific characteristics of the target domain. In addition, DTL1 shows a significant boost in accuracy and recall, which is good for tasks where both precise classification and sensitivity are critical. As a result, Strategy 1 with DTL1 is particularly effective in scenarios where maximizing detection accuracy and minimizing false positives are important.

Table 14: Improvement Metrics for Different Models under Various Scenarios

Metric	Objective	Scenario	Best Model	Improvement (%)
Accuracy	2	DTL1	BiLSTM	13.90
Recall	1	DTL2	BiLSTM	21.48
Precision	2	DTL3	Inception	10.09
F1-score	2	DTL1	BiLSTM	12.22

Based on the data reported in Table 14, we plotted the (best iteration out of 5) BiLSTM and Inception models in terms of Training and Valida-

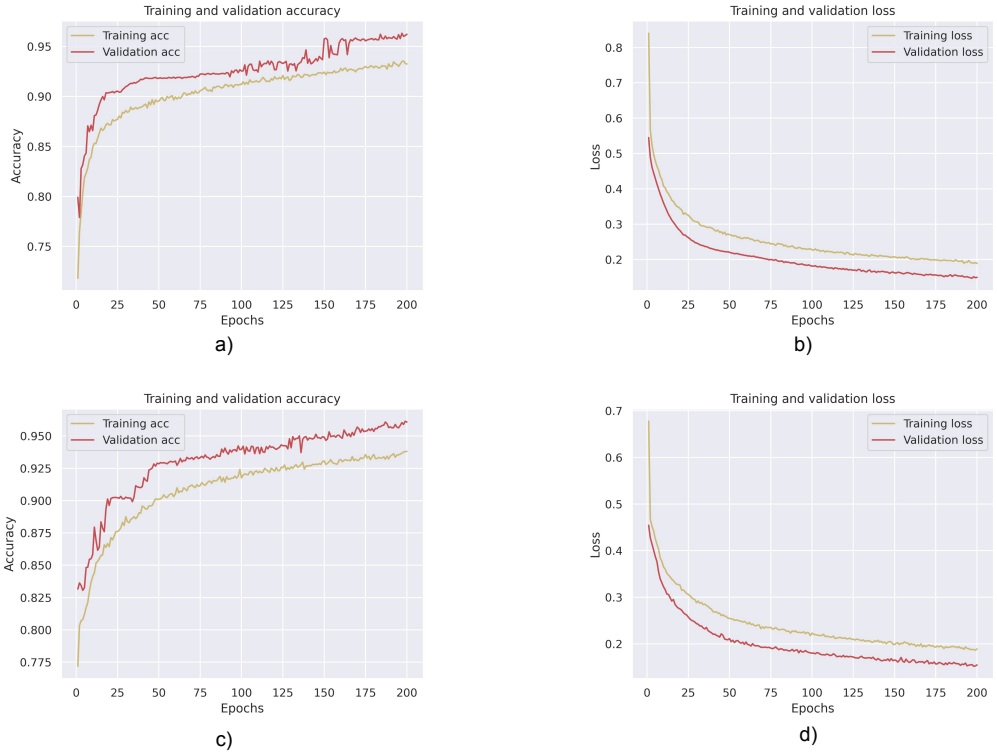

Figure 3: Training and Validation Performance (%) based on 5G-NIDD dataset with DTL1 a) BiLSTM Accuracy Objective 1. b) BiLSTM Loss Objective 1. c) BiLSTM Accuracy Objective 2. d) BiLSTM Loss Objective 2.

tion accuracy and Training and Validation loss values for DTL1, DTL2, and DTL3. These values are depicted in Figures 3, 4, and 5, respectively.

The BiLSTM model's performance on the 5G-NIDD dataset using the DTL1 scenario is shown in Figure 3. The model was evaluated for two objectives. Regarding Objective 1, the training and validation accuracy (subplot a) begin at roughly 72% and 79%, respectively, and gradually increase to around 94% and 96% after 200 epochs. The training and validation loss (subplot b) decrease significantly from 80% to around 19% and 14%, respectively. Regarding Objective 2, the training and validation accuracy (subplot c) first start at roughly 77% and 83%, respectively, and then increase to around 93% and 97%. The training and validation loss (subplot d) exhibit a significant decrease, dropping from 70% to approximately 20% and 15%.

Figure 4 depicts the performance of the BiLSTM model on the 5G-NIDD dataset using DTL2 with two targets. Regarding Objective 1, the training

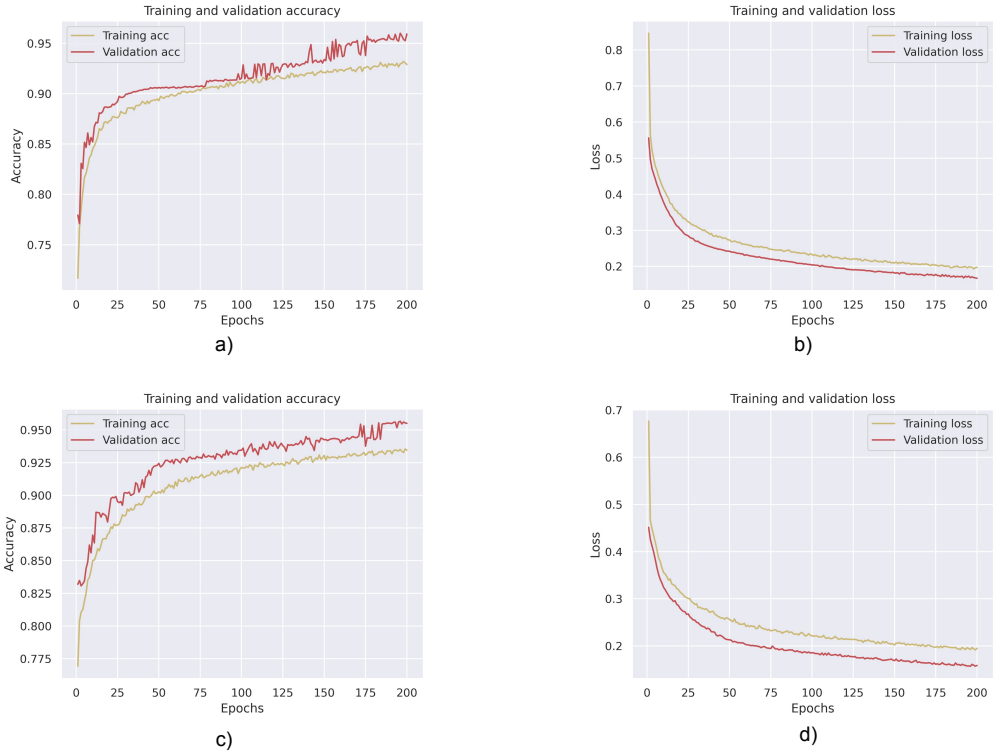

Figure 4: Training and Validation performance (%) based on 5G-NIDD dataset with DTL2 a) BiLSTM Accuracy Objective 1. b) BiLSTM Loss Objective 1. c) BiLSTM Accuracy Objective 2. d) BiLSTM Loss Objective 2.

and validation accuracy (subplot a) exhibit notable enhancement, reaching a plateau at around 93% and 96%, respectively, after 200 epochs. In a similar manner, the training and validation loss (shown in subplot b) exhibit a quick decline from 80% to approximately 20% for training and 18% for validation. Regarding Objective 2, the initial training and validation accuracy (subplot c) are approximately 76% and 83%, respectively. These values thereafter increase to roughly 93% and 96%. The training and validation loss (subplot d) decrease from 70% to around 20% and 14%, respectively.

The narrow discrepancy between the training and validation measures in both objectives demonstrates the robust performance of the model and indicates successful learning and high generalization without notable overfitting in both Figures 4 and 5. The results highlight the reliability and effectiveness of the BiLSTM model for the 5G-NIDD dataset and show its potential usefulness in real-world situations.

The figure 5 demonstrates the training and validation performance of the Inception model on the 5G-NIDD dataset with DTL3, assessed based on two objectives. Regarding Objective 1, the Inception model's training and validation accuracy (subplot a) demonstrates a swift enhancement, reaching a plateau of around 93% and 95%, respectively, after 140 epochs. Similarly, the training and validation loss (subplot b) reduce from 45% to around 20% and 15%, respectively. In Objective 2, the BiLSTM model's training and validation accuracy (subplot c) increases from roughly 78% to around 93% and 95%, respectively, during 140 epochs. The training and validation loss (subplot d) decrease from 45% to around 17% and 14%, respectively. Both objectives demonstrate a negligible disparity between the measures of training and validation and suggest the successful acquisition of knowledge and robust generalization without any notable overfitting. The continuous performance of the Inception model on the 5G-NIDD dataset showcases its strength and dependability and indicates its suitability for practical use.

5. Conclusion and Future Direction

This paper focuses on employing DTL to detect DDoS attacks in 5G networks by introducing a novel technique. Using a methodical and robust approach is of utmost importance if we intend to overcome the novel and complex challenges the 5G environment presents in terms of cybersecurity. It would not suffice for this approach to only assess the performance of a model in controlled environments; rather, it must be able to gauge the ability of the model to adjust to changing patterns of data. The approach presented here consists of two phases, namely the first phase in which the source domain is used for training and the second phase, where what was learned from the source domain is applied to the target domain through different DTL scenarios. In both phases, BiLSTM and CNN-based algorithms including CNN, ResNet, and Inception are used as the underlying learning components.

Finally, a full analysis of different models using various DTL strategies and scenarios on the 5G-NIDD dataset as a baseline shows the unique benefits of each model in improving certain performance metrics. The BiLSTM model demonstrates the best improvements over baseline across most metrics, particularly improving 13.90%, 21.48%, and 12.22% in accuracy, recall, and F1-score, respectively, which highlights its capability to adapt effectively to complex datasets and achieve balanced performance. In addition, the Inception model leads in precision and showcases its strength in minimizing

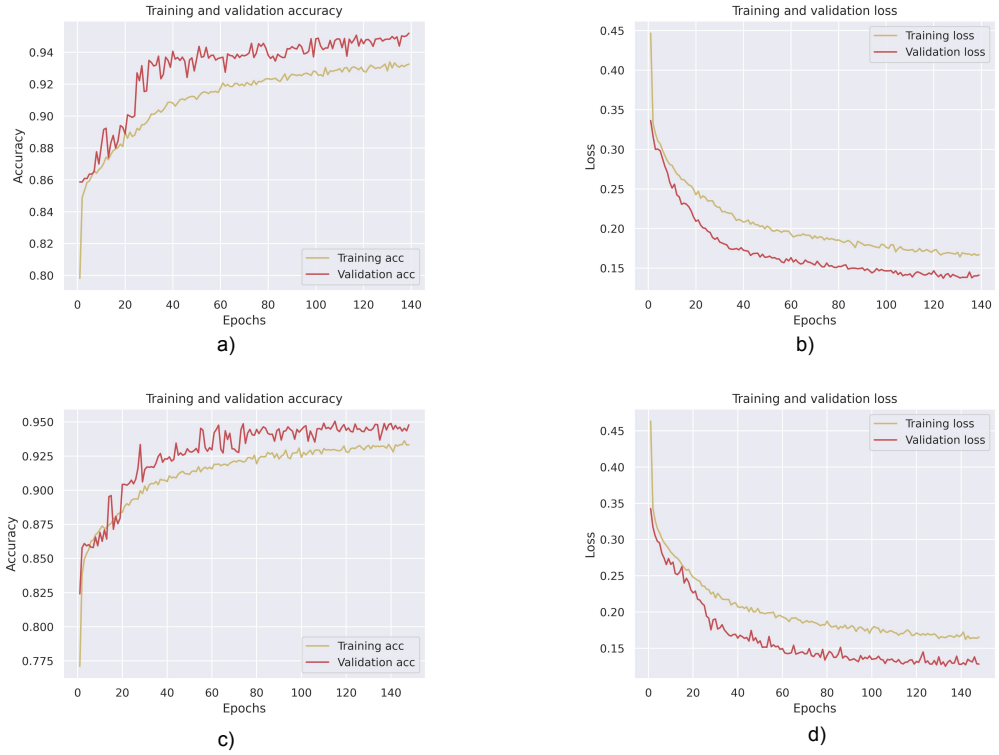

Figure 5: Training and Validation performance (%) based on 5G-NIDD dataset with DTL3 a) Inception Accuracy Objective 1. b) Inception Loss Objective 1. c) Inception Accuracy Objective 2. d) Inception Loss Objective 2.

false positives, which is crucial for applications where precision is important. This analysis gives us useful information for choosing the best model and DTL strategy for each task, which leads to better model performance and reliability in real-world situations.

The methodologies described in this paper can be expanded to identify and possibly mitigate additional network threats, apart from DDoS attacks. Exploring its suitability for different cybersecurity situations shows great potential for 5G-based networks. For this purpose, our next step will be to mitigate not only DDoS attacks but also other types of attacks. Furthermore, we will implement DTL strategies across various network slices to evaluate the effectiveness of our methodologies in 5G network slicing, given that we generated our dataset using a slice-based approach.

Moreover, we intend to explore additional DL models, such as MLP, UNet, DenseNet, and Hourglass networks, to enhance our methodologies further.

We will also investigate the optimal number of layers to freeze and the appropriate layers to add by employing network architecture search techniques. Extending our approach to other cybersecurity challenges within 5G-based networks will enable us to develop more robust and adaptable mechanisms for emerging new threats.

Acknowledgement

This work was supported in part by funding from the Innovation for Defence Excellence and Security (IDEaS) program from the Department of National Defence (DND).

References

[1] A. Ghosh, A. Maeder, M. Baker, D. Chandramouli, 5g evolution: A view on 5g cellular technology beyond 3gpp release 15, IEEE access 7 (2019) 127639–127651.

[2] G. C. Amaizu, C. I. Nwakanma, S. Bhardwaj, J.-M. Lee, D.-S. Kim, Composite and efficient ddos attack detection framework for b5g networks, Computer Networks 188 (2021) 107871.

[3] M. S. Khan, B. Farzaneh, N. Shahriar, N. Saha, R. Boutaba, Slicesecure: Impact and detection of dos/ddos attacks on 5g network slices, in: 2022 IEEE Future Networks World Forum (FNWF), IEEE, 2022, pp. 639–642.

[4] J. Rodriguez, Fundamentals of 5G mobile networks, John Wiley & Sons, 2015.

[5] K. Weiss, T. M. Khoshgoftaar, D. Wang, A survey of transfer learning, Journal of Big data 3 (2016) 1–40.

[6] C. T. Nguyen, N. Van Huynh, N. H. Chu, Y. M. Saputra, D. T. Hoang, D. N. Nguyen, Q.-V. Pham, D. Niyato, E. Dutkiewicz, W.-J. Hwang, Transfer learning for wireless networks: A comprehensive survey, Proceedings of the IEEE 110 (8) (2022) 1073–1115.

[7] J. He, Y. Tan, W. Guo, M. Xian, A small sample ddos attack detection method based on deep transfer learning, in: 2020 International Conference on Computer Communication and Network Security (CCNS), IEEE, 2020, pp. 47–50.

[8] S. Rameshkumar, R. Ganesan, A. Merline, Progressive transfer learning-based deep q network for ddos defence in wsn., Computer Systems Science & Engineering 44 (3) (2023).

[9] E. Rodríguez, P. Valls, B. Otero, J. J. Costa, J. Verdú, M. A. Pajuelo, R. Canal, Transfer-learning-based intrusion detection framework in iot networks, Sensors 22 (15) (2022) 5621.

[10] B. Farzaneh, N. Shahriar, A. H. Al Muktadir, M. S. Towhid, Dtl-ids: Deep transfer learning-based intrusion detection system in 5g networks, in: 2023 19th International Conference on Network and Service Management (CNSM), IEEE, 2023, pp. 1–5.

[11] M. S. Khan, B. Farzaneh, N. Shahriar, M. M. Hasan, DoS/DDoS Attack Dataset of 5G Network Slicing (September 25 2023). doi:10.21227/32k1-dr12.

[12] S. Samarakoon, Y. Siriwardhana, P. Porambage, M. Liyanage, S.-Y. Chang, J. Kim, J. Kim, M. Ylianttila, 5g-nidd: A comprehensive network intrusion detection dataset generated over 5g wireless network, arXiv preprint arXiv:2212.01298 (2022).

[13] L. P. Joseph, R. C. Deo, R. Prasad, S. Salcedo-Sanz, N. Raj, J. Soar, Near real-time wind speed forecast model with bidirectional lstm networks, Renewable Energy 204 (2023) 39–58.

[14] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects, IEEE transactions on neural networks and learning systems 33 (12) (2021) 6999–7019.

[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the

IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[17] M. Stahlke, T. Feigl, M. H. C. García, R. A. Stirling-Gallacher, J. Seitz, C. Mutschler, Transfer learning to adapt 5g ai-based fingerprint localization across environments, in: 2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring), IEEE, 2022, pp. 1–5.

[18] B. Yang, O. Fagbohungbe, X. Cao, C. Yuen, L. Qian, D. Niyato, Y. Zhang, A joint energy and latency framework for transfer learning over 5g industrial edge networks, IEEE Transactions on Industrial Informatics 18 (1) (2021) 531–541.

[19] Z. Lv, R. Lou, A. Kumar Singh, Q. Wang, Transfer learning-powered resource optimization for green computing in 5g-aided industrial internet of things, ACM Transactions on Internet Technology (TOIT) 22 (2) (2021) 1–16.

[20] J. Guan, J. Cai, H. Bai, I. You, Deep transfer learning-based network traffic classification for scarce dataset in 5g iot systems, International Journal of Machine Learning and Cybernetics 12 (11) (2021) 3351–3365.

[21] R. W. Coutinho, A. Boukerche, Transfer learning for disruptive 5g-enabled industrial internet of things, IEEE Transactions on Industrial Informatics 18 (6) (2021) 4000–4007.

[22] T. Mai, H. Yao, N. Zhang, W. He, D. Guo, M. Guizani, Transfer reinforcement learning aided distributed network slicing optimization in industrial iot, IEEE Transactions on Industrial Informatics 18 (6) (2021) 4308–4316.

[23] C. Benzaïd, T. Taleb, A. Sami, O. Hireche, A deep transfer learning-powered edos detection mechanism for 5g and beyond network slicing, in: GLOBECOM 2023-2023 IEEE Global Communications Conference, IEEE, 2023, pp. 4747–4753.

[24] C. Benzaïd, T. Taleb, A. Sami, O. Hireche, Fortisedos: A deep transfer learning-empowered economical denial of sustainability detection framework for cloud-native network slicing, IEEE Transactions on Dependable and Secure Computing (2023).

[25] Free5gc - open-source 5g core network, https://www.free5gc.org/, accessed: May 24, 2023.

[26] Ali Gungr, UERANSIM: 5G Core Network Simulator, https://github.com/aligungr/UERANSIM, accessed on May 24, 2023.

[27] A. S. Mamolar, Z. Pervez, J. M. A. Calero, A. M. Khattak, Towards the transversal detection of ddos network attacks in 5g multi-tenant overlay networks, Computers & Security 79 (2018) 132–147.

[28] M. A. S. Monge, A. H. González, B. L. Fernández, D. M. Vidal, G. R. García, J. M. Vidal, Traffic-flow analysis for source-side ddos recognition on 5g environments, Journal of Network and Computer Applications 136 (2019) 114–131.

[29] Y. Kim, Y. Kim, H. Kim, A comparison experiment of binary classification for detecting the gtp encapsulated iot ddos traffics in 5g network, Journal of Internet Technology 23 (5) (2022) 1049–1060.

[30] M. Guşatu, R. F. Olimid, Improved security solutions for ddos mitigation in 5g multi-access edge computing, in: International Conference on Information Technology and Communications Security, Springer, 2021, pp. 286–295.

[31] N. A. E. Kuadey, G. T. Maale, T. Kwantwi, G. Sun, G. Liu, Deepsecure: Detection of distributed denial of service attacks on 5g network slicing—deep learning approach, IEEE Wireless Communications Letters 11 (3) (2021) 488–492.

[32] A. Thantharate, R. Paropkari, V. Walunj, C. Beard, P. Kankariya, Secure5g: A deep learning framework towards a secure network slicing in 5g and beyond, in: 2020 10th annual computing and communication workshop and conference (CCWC), IEEE, 2020, pp. 0852–0857.

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

[34] B. Bousalem, V. F. Silva, R. Langar, S. Cherrier, Ddos attacks detection and mitigation in 5g and beyond networks: A deep learning-based approach, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE, 2022, pp. 1259–1264.

[35] B. Bousalem, V. F. Silva, R. Langar, S. Cherrier, Deep learning-based approach for ddos attacks detection and mitigation in 5g and beyond mobile networks, in: 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), IEEE, 2022, pp. 228–230.

[36] S. Sheikhi, P. Kostakos, Ddos attack detection using unsupervised federated learning for 5g networks and beyond, in: 2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), IEEE, 2023, pp. 442–447.

[37] A. S. Mamolar, P. Salvá-García, E. Chirivella-Perez, Z. Pervez, J. M. A. Calero, Q. Wang, Autonomic protection of multi-tenant 5g mobile networks against udp flooding ddos attacks, Journal of Network and Computer Applications 145 (2019) 102416.

[38] D. Sattar, A. Matrawy, Towards secure slicing: Using slice isolation to mitigate ddos attacks on 5g core network slices, in: 2019 IEEE Conference on Communications and Network Security (CNS), IEEE, 2019, pp. 82–90.

[39] H. Bisht, M. Patra, S. Kumar, Detection and localization of ddos attack during inter-slice handover in 5g network slicing, in: 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), IEEE, 2023, pp. 798–803.

[40] H. Kheddar, Y. Himeur, A. I. Awad, Deep transfer learning for intrusion detection in industrial control networks: A comprehensive review, Journal of Network and Computer Applications 220 (2023) 103760.

[41] P. Z. Ramirez, A. Tonioni, S. Salti, L. D. Stefano, Learning across tasks and domains, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 8110–8119.

[42] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on knowledge and data engineering 22 (10) (2009) 1345–1359.

[43] Y. Lu, Z. Tian, R. Zhou, W. Liu, A general transfer learning-based framework for thermal load prediction in regional energy system, Energy 217 (2021) 119322.

[44] M. S. Khan, Detection of dos and ddos attacks on 5g network slices using deep learning approach (2023).

[45] hping3 man page, `https://linux.die.net/man/8/hping3`, accessed: May 24, 2023.

[46] Applications — research — canadian institute for cybersecurity — unb, `https://www.unb.ca/cic/research/applications.html.`, accessed: May 24, 2023.

[47] Q. Zhou, D. Pezaros, Evaluation of machine learning classifiers for zero-day intrusion detection–an analysis on cic-aws-2018 dataset, arXiv preprint arXiv:1905.03685 (2019).

[48] E. Berei, M. A. Khan, A. Oun, Machine learning algorithms for dos and ddos cyberattacks detection in real-time environment, in: 2024 IEEE 21st Consumer Communications & Networking Conference (CCNC), IEEE, 2024, pp. 1048–1049.

[49] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, Deep transfer learning for iot attack detection, IEEE Access 8 (2020) 107335–107344.

[50] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data mining and knowledge discovery 33 (4) (2019) 917–963.

[51] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International joint conference on neural networks (IJCNN), IEEE, 2017, pp. 1578–1585.

[52] S. Kornblith, J. Shlens, Q. V. Le, Do better imagenet models transfer better?, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 2661–2671.

[53] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE

conference on computer vision and pattern recognition, 2018, pp. 8697–8710.

[54] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, B. Xu, Attention-based bidirectional long short-term memory networks for relation classification, in: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers), 2016, pp. 207–212.

[55] M. F. Aslan, M. F. Unlersen, K. Sabanci, A. Durdu, Cnn-based transfer learning–bilstm network: A novel approach for covid-19 infection detection, Applied Soft Computing 98 (2021) 106912.

[56] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A survey on deep transfer learning, in: Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27, Springer, 2018, pp. 270–279.

[57] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, Advances in neural information processing systems 27 (2014).

[58] M. Raghu, C. Zhang, J. Kleinberg, S. Bengio, Transfusion: Understanding transfer learning for medical imaging, Advances in neural information processing systems 32 (2019).

[59] F. Yu, X. Xiu, Y. Li, A survey on deep transfer learning and beyond, Mathematics 10 (19) (2022) 3619.

[60] S. Baldwin, Compute canada: advancing computational research, in: Journal of Physics: Conference Series, Vol. 341, IOP Publishing, 2012, p. 012001.