Effective Multitask Deep Learning for IoT Malware Detection and Identification Using Behavioral Traffic Analysis

Sajid Ali, Omar Abusabha, Farman Ali, Muhammad Imran, *Member, IEEE*, and Tamer Abuhmed

***Abstract*—Despite the benefits of the Internet of Things (IoT), the growing influx of IoT-specific malware coordinating large-scale cyberattacks via infected IoT devices has created a substantial threat to the Internet ecosystem. Assessing IoT systems' security and developing mitigation measures to prevent the spread of IoT malware is therefore critical. Furthermore, for training and testing the fidelity of cyber security-based Machine Learning (ML) and Deep Learning (DL) approaches, the collection and exploration of information from multiple sources from the IoT are crucial. In this regard, we propose a multitask DL model for detecting IoT malware. Our proposed Long Short-Term Memory (LSTM) based model efficiently performs two tasks: 1) determination of whether the provided traffic is benign or malicious, and 2) determination of the malware type for identifying malicious network traffic. We used large-scale traffic data of 145 *.pcap* files of benign and malicious traffic collected from 18 different IoT devices. We performed a time-series analysis on the packets of traffic flows, which were then used to train the proposed model. The features extracted from the dataset were categorized into three modalities: flow-related, traffic flag-related, and packet payload-related features. A feature selection approach was employed at the feature and modality levels, and the best modalities and features were utilized for performance enhancement. For tasks 1 and 2 and multitask classification, the flow-related and flag-related modalities showed the best testing accuracies of 92.63%, 88.45%, and 95.83%, respectively.**

***Index Terms*—Multitask deep learning, multimodal learning, Cybersecurity, IoT malware detection, malware identification, and heterogeneity traffic analysis.**

Manuscript received 1 May 2022; revised 7 August 2022; accepted 16 August 2022. Date of publication 23 August 2022; date of current version 6 July 2023. This research was supported by the MSIT (Ministry of Science and ICT), South Korea, under the ICT Creative Consilience Program (IITP-2021-2020-0-01821) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C1011198). The associate editor coordinating the review of this article and approving it for publication was N. Kumar. *(Corresponding author: Tamer Abuhmed.)*

Sajid Ali is with the Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: sajidali@skku.edu).

Omar Abusabha and Tamer Abuhmed are with the College of Computing and Informatics, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: 404970@g.skku.edu; tamer@skku.edu).

Farman Ali is with the Department of Software, Sejong University, Seoul 05006, South Korea (e-mail: farmankanju@sejong.ac.kr).

Muhammad Imran is with the Institute of Innovation, Science and Sustainability, Federation University Australia, Brisbane, QLD 4000, Australia (e-mail: dr.m.imran@ieee.org).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNSM.2022.3200741, provided by the authors.

Digital Object Identifier 10.1109/TNSM.2022.3200741

I. Introduction

INTERNET of Things (IoT) is transforming the connected world with automobiles, smart homes and cities, manufacturing, healthcare systems, retail, space applications, and cyber-physical systems because the number of portable Internet-connected devices is increasing continuously [1]. As a result, with the introduction of new IoT devices, technical advancement and the manufacturing simplicity of these intelligent devices continue to improve. Meanwhile, the creation of a slew of new security considerations is required, according to the 2020 IoT threat report [2]. The Mirai botnet, for example, is used in the well-known cyberattacks on Dyn, which resulted in one of the most significant Distributed Denial-of-Service (DDoS) attacks ever recorded on the Internet [3]. More importantly, the availability of the Mirai source code accelerated the development of powerful and sophisticated Mirai-like malware, such as Satori [4], Hajime [5], and BrickerBot [6], to mention a few. As a result, academics have sought to investigate complex solutions to address the security problem in recent years. However, the lack of empirical data regarding current IoT malware and understanding of the behavioral features of malware-infected devices make the implementation of complex solutions controversial in IoT. Various IoT-specific honeypots have been set up to collect precise information on existing IoT malware [7].

Based on the literature review conducted in this study, we identified two distinct problems: IoT security against malware attacks [8], [9] and classification of IoT malware based on generated traffic [10]. The primary goal is to protect devices from malware attacks. However, due to the emergence of highly sophisticated ransomware, devices cannot be fully secured. It is more likely to target a particular type of malware at a given moment. As a result, it may be feasible to classify different malware types, shutting down services on the targeted malware rather than the entire system. We combined the objective of adequately protecting the IoT device twofold from the security and malware classification perspective. Thus, we proposed a multitask classification model to address both problems simultaneously.

1) IoT Security. Several studies have recognized the relevance of IoT vulnerabilities, emphasizing the need for enhanced intrusion detection algorithms based on the anomaly, signature, and specification [11]. Validating the fidelity of novel intrusion detection models remains challenging because

(1) IoT devices deliver markedly less traffic than workstations and servers, and (2) the persistent application of advanced IoT devices implies the development of innovative types of attacks and traffic [12]. The first objective is to identify the incoming traffic as benign or malware. Therefore, **Task 1** is a binary classification.

2) Malware Classification. Determining helpful and trustworthy features for traffic monitoring remains challenging. Therefore, the two primary types of network traffic classification [10]: (a) *flow-based classification* uses qualities including flow bytes per second and duration per flow, and (b) *packet-based classification* uses values including size and inter-packet duration. We divided the dataset into three modalities: flow, flag, and packet. **Task 2** is to identify the type of malware in each modality.

Several Machine Learning (ML) approaches with flow-based [13], [14], and packet-based [15], [16] features have been developed to identify IoT traffic accurately. However, classifiers based on ML frequently need domain expertise and time-consuming feature extraction and assessment activities. As IoT malware evolves, such customized features may be ineffective in detecting and classifying new malware families [17]. Therefore, recent work proposes the utilization of deep learning (DL)-based malware classification algorithms to address the limitations of ML. These studies propose the lowering of the cost of artificial feature generation while learning features directly from raw data without requiring extra feature engineering [18], [19]. Despite the human-level performance of DL, most state-of-the-art techniques still learn static or dynamic features from a single representation of the malware data, thereby restricting the learning process while disregarding the advantages of employing different representations of the target data. Therefore, advancing a plausible dataset is crucial for building robust models and intrusion detection tools to identify and probe cyber-attacks.

The heterogeneity of datasets aids security researchers in discriminating between normal and malicious incoming traffic from IoT devices to address the difficulties mentioned above. We define the heterogeneity of the datasets along the following two axes: (a) *the network axis* - the dataset should be produced on a network that includes a variety of devices and network technologies, and (b) *the dataset axis* - the dataset should reflect the diverse traffic data and cyberattacks. In this regard, we propose a Long Short-Term Memory (LSTM) DL-based multitask intrusion detection technique to solve previously described constraints by utilizing the multimodal dataset. We illustrate how dataset heterogeneity increases the learning rate of the DL-based approach as a multitask model and explain the importance of dataset heterogeneity for successful intrusion detection in IoT. Our special considerations include various IoT devices and attack kinds. The contributions of the current study are listed below:

- We propose a multitask LSTM-based DL model for two classification tasks: **task 1** determines if the provided traffic is benign or malicious, and **task 2** specifies the malware type.
- We investigate the functionality of the LSTM DL model to predict two tasks using heterogeneous time series data. A three-step time series network traffic data is prepared to evaluate the proposed LSTM-based multitask model effectively. After a careful review of the literature and consultation with specialists in the field, we divide the features into three modalities: flow-related, flag-related, and packet payload-related. The feature selection approach is employed at the modality level, and the selected features from each modality are merged to enhance the performance.
- We conducted several single-task and multitasking experiments to assess the performance of the proposed model using: (1) a class imbalance technique, (2) a feature selection method, and (3) the modality fusion of the time series network traffic data.
- Because there are various devices, protocols, and control interfaces, the rapid increase in IoT devices has brought new issues for device identification. Furthermore, the IP or MAC addresses typical IoT devices used to identify one another in a network are vulnerable to spoofing. We recognize the IoT device based on the network traffic to address the above problem. We evaluate the proposed model by grouping various IoT devices from different sources.
- We generated the time-domain features using the CICFlowMeter [20]. It is an open-source utility that extracts features from *.pcap* files and creates bidirectional (forward and backward) flow.

Organization: The rest of the paper is organized as follows. Section II discusses the related work at the different axes of the IoT malware classification. Section III discusses the preliminaries of the LSTM internal architecture of the cell and the proposed multitasking model. Section IV explains the proposed multitasking DL model with all the modules, such as dataset description, inclusion criterion, experimental environment, and evaluation matrices. Section V describes the proposed multitasking DL-based model with all the modules, such as dataset preprocessing and methodologies. Section VI discusses the experimental results and analysis. Section VII concludes our study and discusses future directions.

II. Literature Review

A literature review of four types of studies on the IoT network is presented in this section. We present an overview of the existing datasets and progress toward creating new databases. We then explore statistical, ML-based, and DL-based strategies for classifying IoT traffic. We also discuss studies on IoT intrusion detection.

IoT DATASETS: Most existing intrusion detection technologies aimed at IoT devices are examined and analyzed by modeling attacks using relatively old datasets [21]. For example, the NSL-KDD [22], KDD CUP [23], UNSW-NB15 [24], CTU-13 [25], and CICIDS [26] datasets do not adequately reflect the diverse characteristics of modern IoT devices, which often comprise a wide range of protocols and standards. A few

pioneer IoT datasets, such as the IoT network intrusion [27], N-BaIoT [28], ToN_IoT [12], and IoT-23 [29], attempted to overcome this deficiency and served as a baseline for comparing different intrusion detection approaches. However, the variety of IoT devices and the type of attacks that can be investigated using these datasets are very restricted. Furthermore, several datasets were constructed from a single data source. For instance, the IoT network intrusion data include *.pcap* files, whereas N-BaIot data comprises sensor measurements.

STATISTICAL METHODS: Since the inception of digital networks, automating intrusion detection has been a key research topic. The importance and need for datasets on real-world attacks and standard traffic are commonly agreed upon. Many such datasets in the public domain have been designed to identify anomalies in general network traffic (e.g., NSL-KDD [22] and CTU-13 [25]), and only a few of them have been built primarily for IoT networks.

Static malware analysis approaches are used to better understand IoT malware. For instance, BotHunter [30] was developed to identify the infection and coordination communication after a successful malware attack. A similar method, BotSniffer [31] focuses on detecting botnet command and control channels. BotHunter and BotSniffer test on their honeynets or traces created by malware binaries being executed, which, however, are not publicly available. This highlights the absence of appropriate comparisons for botnet identification algorithms due to a lack of publicly available botnet datasets. A method for malware detection by graph's spectral heat and wave signatures is also presented [32].

ML-BASED METHODS: While extracting features to improve threat mitigation by building effective malware classification techniques based on ML or DL algorithms, ML-based algorithms can distinguish between normal and malicious traffic patterns using relevant datasets [33], [34]. *Supervised ML* is broadly classified as support vector machine (SVM)-based classification, ensemble-learning, and rule-based approaches that learn from a labeled dataset comprising regular traffic or attacks [35]. SVM is commonly utilized in various security applications because of its outstanding generalization performance [36]. However, Shafiq *et al.* [37] examined 44 features using the Bot-IoT dataset to establish a framework model for evaluating attack detection methods. According to their methodology, the best algorithm was based on Naive Bayes. Hasan *et al.* [38] presented a multi-layer model. The first layer detects anomalous behavior, while the second layer determines the attack type the device is experiencing. In their research, the random forest classifier has a maximum accuracy of 99.40% using the DS2OS dataset. Various features from malware traffic using static and dynamic analyses are evaluated on seven ML models [39]. Furthermore, android malware features are extracted using the famous algorithms and built three ML models [40].

Unsupervised ML methods learn from unlabeled datasets to determine abnormalities. For instance, Lagraa *et al.* [41] devised a BotGM technique for determining the most distinct graphs using an unsupervised approach. Bhatia *et al.* [42] presents an auto-encoder method based on unsupervised learning for detecting known and unknown abnormalities in IoT.

DL-BASED METHODS: Several DL algorithms have been presented for malware traffic monitoring and classification. For example, Gao *et al.* [43] used deep belief networks. Shone *et al.* [44] proposed a novel approach by integrating the non-symmetric deep auto-encoder with a random forest called a sparse auto-encoder. Both used data of hand-drawn flow features as input. Bendiab *et al.* [45] utilized recurrent neural networks to learn the temporal characteristics of the IoT traffic, which are converted into a series of characters. In contrast, Shire *et al.* [46] employed the convolutional neural network MobileNet to analyze IoT malware traffic. However, the two techniques are only evaluated on tiny datasets (approximately 100 samples), limiting neural network training alternatives. Baptista *et al.* [47] used self-organizing incremental neural networks to detect malware executables rather than IoT traffic. They used a residual neural network to train and test additional malware samples and valid *.pcap* files.

OTHER METHODS: Entropy measures are used as an alternative to ML- and DL-based methods. By evaluating the difference between the profiles of typical traffic and incoming flow data, Francois *et al.* [48] detected large-scale abnormalities in network traffic. Using Markov Chains to simulate the multiple states in the command and control channel, Garcia *et al.* [49] provided a behavioral botnet detection approach. Singh *et al.* [50] focused on finding bot-infected devices at an enterprise level by monitoring an entire DNS activity every hour. Dib *et al.* [51] used several string-based features for malware classification and clustering.

III. Problem Formulation and Preliminaries

The value of one feature at a specified instant in multivariate time-series data analysis is usually determined based on its previous values and the values of other features [52]. For example, let k represent the total number of time-series features such that $\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \ldots, \mathcal{X}_k\}$, the prediction value of $\mathcal{X}_k$ at time t depends on the values of $\mathcal{X}_k$ at the current time step and all previous time steps, i.e., $\{\mathcal{X}_k^t, \mathcal{X}_k^{t-1}, \ldots, \mathcal{X}_k^{t-k}\}$. It depends on $n = k \times t$ historical values of the other features as expressed below,

$$\hat{\mathcal{X}}_1^{t+1} = \mathbf{F}\Big(\mathcal{X}_1^t, \mathcal{X}_2^t, \ldots, \mathcal{X}_k^t; \cdots; \mathcal{X}_1^{t-1}, \mathcal{X}_2^{t-1}, \ldots, \mathcal{X}_k^{t-1}; \cdots; \mathcal{X}_1^{t-n}, \mathcal{X}_1^{t-n}, \ldots, \mathcal{X}_k^{t-n}\Big)$$

where $\mathbf{F}(.)$ denotes the DL-based model used to predict the future time step of $\mathcal{X}$.

In the class prediction process, the LSTM DL model uses the temporal correlation between the historical values of each time-series feature as well as the correlation information between multivariate time-series [53], [54]. The essential architecture of the LSTM employs three significant gating units: ⓐ the input unit $(\mathcal{I}^{t_n})$ for updating, ⓑ the forget unit $(\mathcal{F}^{t_n})$ for maintenance, and ⓒ the output unit $(\mathcal{O}^{t_n})$ for deleting, as shown in Figure 1. The equation of each gating unit

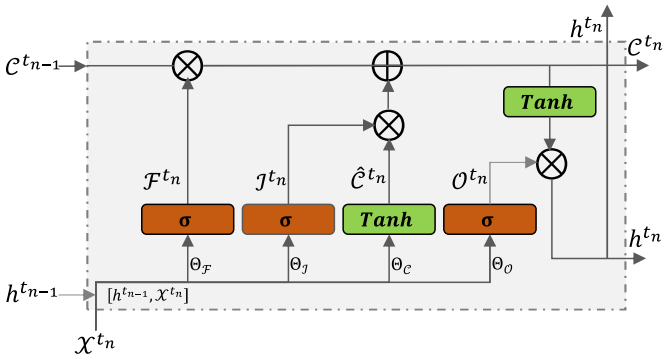

Fig. 1. The internal structure of the LSTM cell. The cell illustrates the three primary gating units.

may be found as follows:

$$\mathcal{I}^{t^n} = \sigma\big(\Theta_{\mathcal{I}} \cdot \big[h^{t_{n-1}}, \mathcal{X}^{t_n}\big] + b_{\mathcal{I}}\big)$$
$$\mathcal{F}^{t^n} = \sigma\big(\Theta_{\mathcal{F}} \cdot \big[h^{t_{n-1}}, \mathcal{X}^{t_n}\big] + b_{\mathcal{F}}\big)$$
$$\mathcal{O}^{t^n} = \sigma\big(\Theta_{\mathcal{O}} \cdot \big[h^{t_{n-1}}, \mathcal{X}^{t_n}\big] + b_{\mathcal{O}}\big)$$

A controlling parameter is assigned to each gating unit. At time instance t_n, $\mathcal{C}^{t_n}$ denotes the current or present status of the cell, $\mathcal{C}^{t_{n-1}}$ represents the previous status, and $\hat{\mathcal{C}}^{t_n}$ stores the updated value of the status of the LSTM cell. Furthermore, h^{t_n} denotes the hidden layer. The terms are expressed as below:

$$\hat{\mathcal{C}}^{t_n} = Tanh\big(\Theta_{\mathcal{C}} \cdot \big[h^{t_{n-1}}, \mathcal{X}^{t_n}\big] + b_{\mathcal{C}}\big)$$
$$\mathcal{C}^{t_n} = \mathcal{F}^{t_n} \otimes \mathcal{C}^{t_{n-1}} \oplus \mathcal{I}^{t_n} \otimes \hat{\mathcal{C}}^{t_n}$$
$$\hat{h}^{t_n} = \mathcal{O}^{t_n} \otimes Tanh\big(\mathcal{C}^{t_n}\big)$$

where the output by the hidden layer's memory cell is $h^{t_{n-1}}$ and the set of weight matrices and biases for the gating units are depicted by Θ_* and b_*, respectively. The standard logistic sigmoid function is σ. The Hadamard product is $\otimes$. The concatenation operation is performed by $\oplus$. The output activation function is *SoftMax* or *Tanh*.

PROPOSED MODEL: Our proposed LSTM network jointly learns the two correlated tasks: ❶ y^1: malware classification, and ❷ y^2: type of IoT device classification. Both these tasks are multi-class classification. The diverse family of IoT network traffic of two multivariate datasets represented as $\mathcal{S} = \{\mathcal{D}^1, \mathcal{D}^2\}$ is considered. Each dataset, $\mathcal{D}^m = \{\mathcal{X}_1^m, \ldots, \mathcal{X}_i^m, \ldots, \mathcal{X}_{\mathcal{N}}^m\}$ has $\mathcal{N}$ IoT devices where each device $\mathcal{X}_i^m$ is a multivariate time-series sequence, $\mathcal{X}_i^m = \{\mathcal{X}_{i1_t}^m, \mathcal{X}_{i2_t}^m, \ldots, \mathcal{X}_{if_t}^m\}$ for $t = (1, \ldots, \mathcal{N})$ time-steps for i^{th} dataset. As a result, every dataset is defined as $\mathcal{D}_i = \{\mathcal{X}_{i1_t}^m, \mathcal{X}_{i2_t}^m, \ldots, \mathcal{X}_{if_t}^m, y_i^1, y_i^2\}$, $i = 1, \ldots, N$, where y_i^1 is the discrete label of malware classes, and y_i^2 is the class label of the IoT devices for the i^{th} instance.

The optimization is simultaneously performed on the shared parameters (Θ^{sh}) based on the specific task (Θ^t) for malware classification and IoT device classification. The hypothesis of each task is parametrized as $f^t(\mathcal{X}, \Theta^{sh}, \Theta^t) : \mathcal{X} \rightarrow y$, and the specific loss functions of both classification tasks are $\mathcal{L}^t(.) : y^t \times y^t \rightarrow R^+$. The multi-objective optimization problem may be expressed to minimize the proposed model loss, as shown below:

$$\min_{\substack{\Theta^{sh} \\ \Theta^1, \Theta^2}} \mathcal{L}\Big(\Theta^{sh}, \Theta^1, \Theta^2\Big) = \min_{\substack{\Theta^{sh} \\ \Theta^1, \Theta^2}} \Big(\hat{\mathcal{L}}^1\Big(\Theta^{sh}, \Theta^1\Big), \hat{\mathcal{L}}^2\Big(\Theta^{sh}, \Theta^2\Big)\Big)$$

where $\hat{\mathcal{L}}^t(.)$ is the loss function for a specific task represented as

$$\hat{\mathcal{L}}^t\Big(\Theta^{sh}, \Theta^t\Big) = \frac{1}{N} \sum_i \mathcal{L}\Big(f^t\Big(\mathcal{X}_i; \Theta^{sh}, \Theta^t\Big), y_i^t\Big)$$

We addressed both tasks in a joint objective function defined as follows, for m is the number of Θ^{st} and Θ_t parameters.

$$\hat{\mathcal{L}}^t\Big(\Theta^{sh}, \Theta^t\Big) = -\left[\frac{1}{N} \sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\right] + \frac{\lambda}{m} \sum_{j=1}^{m} \Theta_j^2$$

where the cross-entropy loss is used for the multi-class classification task with

$$p_i = \frac{1}{1 + exp\big(-W^T \mathcal{X}_i\big)}$$

The term at the end is the regularization term.

IV. Material and Methodology

This section discusses the dataset, preparation methods, and theoretical explanations concerning the techniques and measurements used in this study.

A. Dataset Heterogeneity

The data source $\mathcal{S} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ comprises n different datasets to express the data heterogeneity. Each dataset $\mathcal{D}$ consists of $\mathcal{X}$ multivariate features. In this study, we employed two different datasets.

- D_1 IoT-23 dataset of Avast AIC laboratory, which comprises 20 malware samples from numerous IoT devices and three benign anomaly samples, all recorded between 2018 and 2019 [29].
- D_2 A legitimate IoT network traffic dataset is used, created as a part of the EU CEF VARIoT project under realistic conditions [55].

The datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ are available in the *.pcap* format. The network traffic features from *.pcap* are extracted using a CICFlowMeter tool and saved in *.csv* format. We preferred the CICFlowMeter software because it gave us greater flexibility in selecting user-defined features, adding features, and controlling the flow time-out duration. The CICFlowMeter software creates bidirectional flows, with the forward (source to destination) and backward (destination to source) directions determined by the first packet. Table I shows statistical time-related features determined in the forward and backward directions. Besides the duration, there are six groupings of features, which represents the overall flow period. The first three categories, **-fiat**, **-biat**, and **-flowiat**, are concerned with the forward, backward, and bi-directional flows, respectively. The idle-to-active or active-to-idle states are determined for the

TABLE I
A RANGE OF STATISTICAL TIME-DOMAIN FEATURES IS DEFINED USING THE CICFLOWMETER UTILITY

Feature	Description
duration	The length of time the flow ends.
fiat*	The interval between two packets is transmitted in the forward direction.
biat*	The interval between two packets is transmitted in the backward direction.
flowiat*	The interval between two packets is transmitted in either direction.
active*	The length of time a flow remained active before becoming idle.
idle*	The length of time a flow remained idle before becoming active.
fb_psec	Flow bytes per second.
fp_psce	Flow packets per second.

fiat - Forward Inter Arrival Time, biat - Backward Inter Arrival Time, flowiat - Flow Inter Arrival Time.
∗time-domain features (min, max, mean, std, variance) are computed.

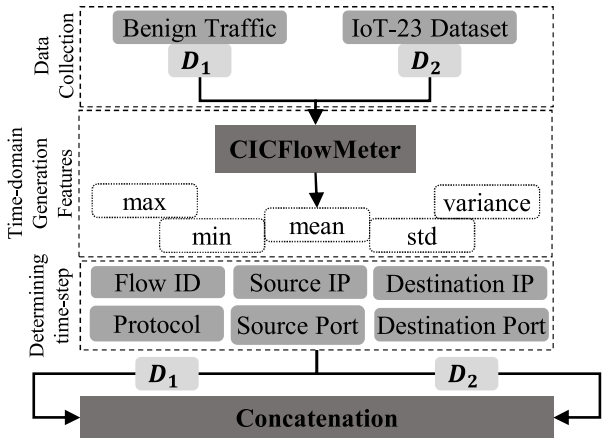

Fig. 2. Depicts the flow of processing both the dataset and the data. The CICFlowMeter is used to extract time-domain features. From each dataset, we computed three-time steps with the same flow ID, source IP, source port, destination IP, destination port, and protocol extracted.

fourth and fifth sets of features, designated **-idle** and **-active**, respectively. The last group, known as the **-psec** feature, is concerned with the size and the number of packets per second.

B. Dataset Inclusion and Preparation

We used the IoT traffic of a heterogeneous dataset in this study. Namely, two datasets gathered from various sources were used. The network (*.pcap*) format is provided for each dataset. The IoT-23 dataset contains 23 captures, with 20 malware and three benign IoT device traffic captures. A distinct malware sample on a Raspberry Pi for each malicious scenario is deployed, utilizing many protocols. The network traffic collected for the benign scenarios came from three IoT devices: a Philips HUE smart LED bulb, an Amazon Echo home intelligent personal assistant, and a Somfy smart door lock. The VARIoT dataset contains 122 benign captures. The network traffic collected for the benign scenarios came from 15 IoT devices. Supplement Table S1 and Table S2 provide a comprehensive list of devices and the encoding of the captured data.

The time-domain features are extracted from the captures (*.pcap*) files using the CICFlowMeter. The network features are stored in the *.csv* format for additional data preparation. The packets with the same flow ID, source IP, source port, destination IP, destination port, and protocol are studied. Three-time steps are determined based on the analysis, which we incorporated into the dataset. It is worth mentioning that identical steps are employed to handle both datasets. In the end, the two datasets are concatenated after extracting the time-domain features using CICFlowMeter, as shown in Figure 2. Supplement Table S3 shows the extracted list of network features.

C. Evaluation Metrics

The following four metrics are used to assess the classification performance of the proposed multitask LSTM model:

Accuracy: The percentage of instances successfully classified out of the total number.

$$Accuracy = \frac{\mathcal{TP} + \mathcal{TN}}{\mathcal{TP} + \mathcal{TN} + \mathcal{FP} + \mathcal{FN}}$$

Precision: The percentage of accurately classified positive instances to all predicted positive instances.

$$Precision = \frac{\mathcal{TP}}{\mathcal{TP} + \mathcal{FP}}$$

Recall: The percentage of accurately classified positive instances to all instances in the actual class.

$$Recall = \frac{\mathcal{TP}}{\mathcal{TP} + \mathcal{FN}}$$

F1-Score: Precision and Recall are weighted into the F1-Score. As a result, false positives and negatives are included while calculating this score.

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

where $\mathcal{TP}$, $\mathcal{TN}$, $\mathcal{FP}$, and $\mathcal{FN}$ represent the true positive, the true negative, the false positive, and the false negative, respectively.

D. Configuration of Model

The Keras framework with TensorFlow as the backend is used to implement the proposed models. The ReLU activation function with binary (benign or malware) cross-entropy loss is utilized for the classification task. In contrast, categorical cross-entropy is employed for the multi-class (type of malware) classification. The proposed binary class, multi-class, and multitask models use an Adam optimizer with 0.001 learning rate optimization. The number of epochs and the training batch size is 100 and 240, respectively. We use a single LSTM layer with 128 hidden units for feature extraction, followed by a l_2-norm and dropout of 0.01 each. Each task layer comprises three dense layers that work with the ReLU function. There are 64, 32, and 32 hidden units for both tasks.

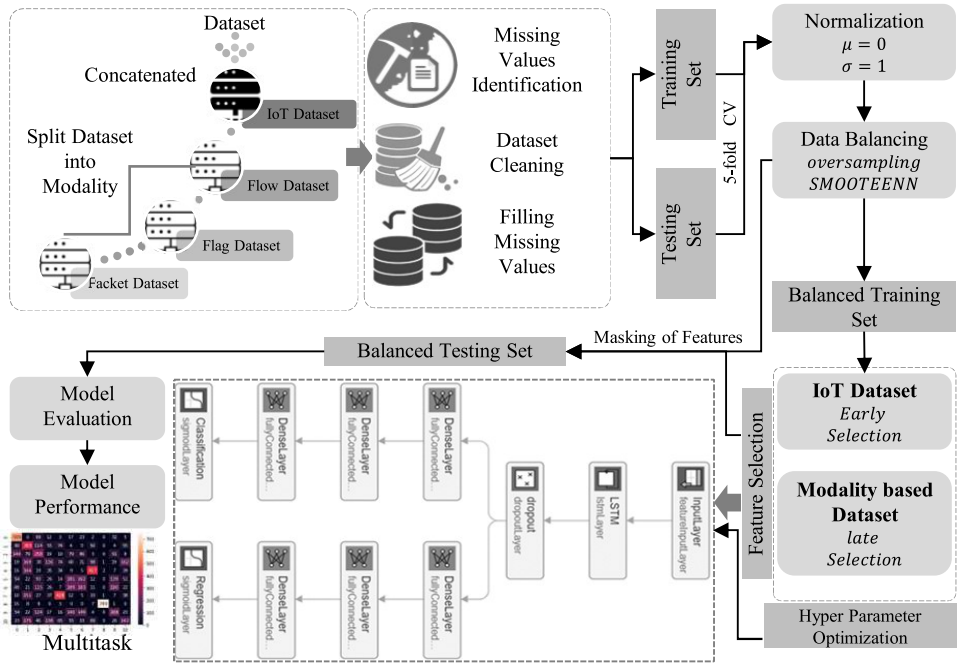

Fig. 3. The proposed framework is presented for the multitask classification. At first, the dataset is divided into three modalities. A number of preprocessing steps are performed before the train and test set. For the model stability, each modality is normalized and performed class imbalanced technique. At last, the model is trained on the selected features and evaluated on the testing set.

EXPERIMENTAL WORKSTATION: The experiments are run on a machine equipped with an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz × 24 with Cuda-10.0 and three GEFORCE GTX TITANx 12 GB GPUs, as well as Python 3.7.7 distributed in Anaconda 4.8.3 (64-bit).

V. Proposed Framework

The proposed framework for IoT malware detection is illustrated in Figure 3. The essential components of the framework are data collection and splitting into modalities, data preprocessing, training and testing sets split, five-fold cross-validation (CV), normalization, class balancing, feature selection, model training, hyperparameter optimization, and model evaluation. In the following sections, we explain all these stages in detail. After the raw data are processed, as discussed in Section IV-B, multiple data preparation steps are performed before training the model. The first step involves splitting the IoT concatenated dataset into the different modalities (flow, flag, and packets). The flow, flag, and packets datasets have 31, 21, and 28 features, respectively. The list of features in each modality is shown in Supplement Table S3. The heterogeneous three-time steps data is subjected to prepossessing to enhance the data quality. The following are some of these steps:

- Missing values for all modalities are handled in the first stage. All features with a missing data percentage exceeding 30% are eliminated. The flow and flag dataset do not have missing elements. There are three features in the packet dataset that are empty. As a result, we eliminate all missing columns and did not require another filling method.
- The dataset is split into a training set (80%) and a testing set (20%). The training set is used to develop and validate the proposed model using the stratified 5-fold CV technique, whereas the testing set is used to determine the model generalization.
- The dataset is also rescaled to make it uniform for model convergence. In the range [0,1], the data are normalized. Outliers are also identified at this point, and each class's average value is used to replace them. The current study normalizes the dataset using the Min-Max approach [56]. The following is the definition of Min-Max normalization:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x_{max} and x_{min} are the maximum and minimum values, respectively. In the range [a,b], the Min-Max normalization translates values x to x'.
- Since DL models are prone to bias for unbalanced datasets, the next step is data balancing. The processed dataset is highly imbalanced, i.e., the benign input traffic instances are 157,593, and malware traffic instances are 8,439 (See Supplement Table S4 for more details). There are a variety of approaches to deal with imbalanced datasets. Commonly used methods are oversampling and undersampling the data using the Synthetic Minority Oversampling Technique (SMOTE) and its variants [57]. After testing multiple SMOTE methods, SMOTE Edited Nearest Neighbor (SMOTEENN) is used as the best fit in the present study to deal with the training set's class imbalance. The testing set is balanced by oversampling.

TABLE II
THE PERFORMANCE COMPARISON BETWEEN TASKS 1 AND 2 USING THE CLASS IMBALANCE TECHNIQUE IS REPORTED ACROSS THE MODALITIES. THE RESULTS OF A FIVE-FOLD CV AND TESTING AVERAGE PERFORMANCE ARE PRESENTED AS MEAN ± SD

Dataset	Cross-validation Performance				Testing Performance			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Task 1 - Binary Classification (Benign or Malware)								
Flow dataset	74.25±3.53	75.62±3.49	77.64±1.87	76.61±3.22	71.49±3.68	72.69±4.76	76.39±3.34	73.97±2.69
Flag dataset	73.58±4.23	76.16±2.56	76.74±1.62	76.04±3.34	71.11±3.79	71.46±3.45	76.62±2.87	73.28±3.41
Packet dataset	72.23±4.28	75.21±2.61	76.90±2.01	77.30±4.15	70.87±3.38	71.03±4.26	76.80±2.26	73.58±3.56
Task 2 - Multi-class Classification (Malware type)								
Flow dataset	69.09±3.92	71.07±3.99	72.17±1.93	69.83±3.04	66.09±4.77	65.41±4.26	70.96±1.60	67.85±2.84
Flag dataset	68.23±4.02	71.98±3.74	71.94±1.17	71.16±3.18	66.26±4.23	66.42±4.61	71.75±1.92	68.92±3.61
Packet dataset	67.75±3.39	71.26±4.04	71.85±1.93	69.89±3.06	64.48±5.73	65.70±5.74	70.92±0.95	67.33±1.68

- In the DL process, feature selection is a crucial aspect of model performance. The present study explored the impact of various feature selection procedures in detail. We test feature selection using two approaches: (a) the entire set of features (concatenated IoT dataset), termed as an early selection of features before splitting the dataset into modalities, and (b) late feature selection, i.e., feature selection method is employed on each modality, and only essential features are combined. We fuse all the modalities and then apply a feature selection method. Since feature selection is applied after fusing the modality, we termed it the late feature selection approach. It is feasible that a feature in one modality depends on the features of another modality. Therefore, we adopted the late feature selection strategy. The recursive XGBoost and SULOV (Searching for Uncorrelated List Of Variables) [58] method reduces features and selects the model's best features. Assume that a dataset $\mathcal{D} = \{(x_i, y_i){:}i = 1, \ldots, n, x_i \in R^m, y_i \in R^m\}$, has n samples with m features. Let $\hat{y}_i$ be defined as the value predicted by the model:

$$\hat{y}_i = \sum_{i=1}^{n} \mathcal{F}_i(x_i) \tag{1}$$

where $\mathcal{F}_i$ represents an independent regression tree, and $\mathcal{F}_i(x_i)$ denotes the prediction score given by the i^{th} tree to the n^{th} sample. The set of functions $\mathcal{F}_i$ in the regression tree model can be learned by minimizing the objective function as follows:

$$\mathcal{O} = \sum_{i=1}^{n} \mathcal{L}(y_i, \hat{y}_i) + \sum_{i=1}^{n} \Omega(\mathcal{F}_i)$$

VI. RESULTS AND DISCUSSION

Based on single and varied fusions of modalities, we report the proposed multitasking LSTM model's performance from various aspects, with the following key objectives: (1) comparison of the binary and multi-class classification as a single-task, (2) comparison between the single-task and multi-tasks, and (3) effectiveness of the feature fusion method. The experimental results are discussed in terms of the mean ± Standard Deviation (SD). The performance of each task across the modality is compared based on the accuracy metric since it reflects other measurement matrices. Readers may check the accompanying tables for further information on the other performance measures.

A. Performance Evaluation With Class Imbalance

Single-task. The class-balanced methodology is used in the first set of experiments. The experiment is designed to evaluate the proposed multitask model of a single-task at a time with class imbalanced and entire features in each modality. The average of five-fold CV and testing performance is listed in Table II for tasks 1 and 2 individually across the modalities. In attaining 74.25% and 69.09% CV accuracy, the flow modality surpassed the other two modalities in both tasks. Flow modality (SD = 3.53%) throughout the training remains stable. The flag dataset achieves 1.35% and 0.78% better accuracy than the packet dataset for tasks 1 and 2, respectively. However, the packet modality is more stable for task 2.

The flow-related dataset achieves a testing accuracy of 71.49% for task 1, whereas the flag-related dataset attains the highest accuracy of 66.26% for task 2. Although the stability of the packet pay-load dataset is less (SD=3.38%), it relatively underperformed.

Multitask: This experiment examines the effectiveness of the proposed model by conducting both the tasks simultaneously using the class imbalance and complete features in each modality. Table III shows the average five-fold CV and testing performance for multitasking across the modalities. For task 1, the flow-related dataset had the highest CV accuracy of 80.33%, while the flag-related dataset had the maximum CV accuracy of 75.7% for task 2. Compared to the single-task, the multitask model achieved over 6% and 5% more accuracy in the case of the flow dataset with 0.09% and 0.61% better stability for tasks 1 and 2, respectively. Similarly, flag-related and packet payload datasets improve significantly during the model training phase.

In the evaluation phase, the flow-related modality achieves the lowest accuracy of 75.77%; but had the highest stability (SD = 4.45%). The flag-related dataset shows the highest testing accuracy of 77.69%. The multitask model achieves an accuracy of over 6.5% in the case of the flag dataset for both the tasks in comparison to the accuracy achieved with the single-task. The flow-related modality improves by 4.28%

TABLE III
THE PERFORMANCE COMPARISON OF MULTITASKING USING THE CLASS IMBALANCE TECHNIQUE IS REPORTED ACROSS THE MODALITIES. THE RESULTS OF A FIVE-FOLD CV AND TESTING AVERAGE PERFORMANCE ARE PRESENTED AS MEAN ± SD

Dataset	Cross-validation Performance				Testing Performance			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Flow dataset								
Task 1	80.44±3.44	79.58±3.24	81.92±1.66	82.41±1.88	75.77±4.45	75.50±4.19	83.57±3.79	77.45±2.50
Task 2	74.44±3.31	75.26±3.76	76.95±2.15	77.96±2.65	72.24±5.18	71.97±4.53	77.45±3.11	73.00±2.38
Flag dataset								
Task 1	79.33±2.71	79.39±3.40	80.75±0.93	82.84±0.90	77.69±5.21	75.56±5.37	84.28±3.49	78.54±1.54
Task 2	75.70±3.54	74.79±2.51	76.38±2.19	78.15±2.13	72.83±5.85	72.60±3.94	76.61±2.98	73.42±2.03
Packet dataset								
Task 1	78.67±3.42	79.47±3.78	79.64±1.22	81.51±1.64	76.94±5.32	74.77±6.30	82.74±2.65	78.96±2.44
Task 2	74.59±2.78	74.91±2.13	76.62±3.55	77.99±2.27	70.73±5.69	72.61±5.31	76.44±4.72	73.26±2.23

TABLE IV
THE PERFORMANCE COMPARISON BETWEEN TASKS 1 AND 2 BY EMPLOYING THE CLASS IMBALANCE AND THE FEATURE SELECTION TECHNIQUES ARE REPORTED ACROSS THE MODALITIES. THE RESULTS OF A FIVE-FOLD CV AND TESTING AVERAGE PERFORMANCE ARE PRESENTED AS MEAN ± SD

Dataset	Cross-validation Performance				Testing Performance			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
task 1 - Binary Classification (Benign or Malware)								
Flow dataset	80.14±3.62	80.88±3.70	81.69±1.43	83.14±2.58	77.18±4.46	76.74±5.49	82.01±2.82	77.37±2.41
Flag dataset	80.07±3.73	80.74±3.99	82.44±1.44	83.04±2.32	77.86±5.36	75.79±5.40	82.11±1.86	77.29±2.50
Packet dataset	81.27±3.82	80.79±3.43	83.26±1.58	83.74±1.93	77.85±5.54	76.70±6.49	82.18±3.14	78.03±3.15
task 2 - Multi-class Classification (Malware type)								
Flow dataset	76.46±5.38	78.83±4.33	78.64±2.09	80.28±3.08	73.77±6.42	73.51±4.01	79.44±1.80	75.69±1.37
Flag dataset	76.23±5.11	79.22±4.56	79.23±1.46	80.07±1.58	74.14±4.80	74.66±3.64	79.91±2.26	75.64±1.62
Packet dataset	76.12±3.41	78.39±3.28	77.64±1.26	78.93±2.90	74.07±6.58	73.61±2.83	79.92±3.30	74.58±1.87

and 6.15% for tasks 1 and 2, respectively. Similarly, the packet payload dataset improves the testing performance significantly.

B. Performance Evaluation With Feature Selection

Single-task: The experiment investigates the role of feature selection in training the proposed model with a class imbalanced technique. The experiment is designed to evaluate the proposed multitask model of the single-task at a time with the class imbalance and selected features in each modality. We employ the recursive XGBoost and SULOV methods for each modality. The summary of the selected features for each modality is in Supplement Table S5. The proposed model is trained and evaluated with the selected features in this experiment. The average five-fold CV and testing performance with the selected features are listed in Table IV for tasks 1 and 2 individually across the modalities. In attaining 80.14% and 76.46% CV accuracy, the flow modality surpasses the other two in both tasks. Throughout the training of task 1, flow modality (SD = 3.62%) remains stable. For task 2, the packet payload modality remains stable with a 3.41% deviation. Compared to the entire features, the flow-related dataset achieves 5.89% and 7.37% better accuracy for tasks 1 and 2, respectively. However, the stability is 0.09% and 1.46% less.

The flag-related dataset achieves a testing accuracy of 77.86% and 74.14% for tasks 1 and 2, respectively. Although the stability of the flow-related dataset is less (SD = 4.46%), it underperformed relatively in both tasks. Compared to entire features with class imbalance, the flag-related dataset achieves 6.75% and 7.88% better testing accuracy for tasks 1 and 2, respectively. However, the packet dataset improves the most by achieving accuracies of 6.98% and 9.59%.

Multitask: The experiment examines the proposed DL-based model by conducting both tasks simultaneously with the class imbalance and selected features in each modality. We employ the same features as described in Supplement Table S5. The average five-fold CV and testing performance for multitasking across the modalities are presented in Table V. For task 1, the flag-related dataset achieves the highest CV accuracy of 82.79%, while the flow-related dataset achieves the maximum CV accuracy of 78.16% for task 2. Compared to single-task, the multitask model achieves over 2.72% and 1.37% more accuracy in the case of the flag dataset with 0.02% and 0.01% lesser stability for tasks 1 and 2 respectively. Similarly, flow-related and packet payload datasets improve significantly during the model training phase. Compared to entire features, the flag-related dataset achieves 3.46% and 1.9% better accuracy for tasks 1 and 2, respectively. However, the stability is 1.04% and 1.58% less. The packet dataset improves the most by achieving an accuracy of 3.5% for task 1, while flow related dataset improves by 3.72% for task 2.

In the evaluation phase, flag-related modality achieved the best accuracy of 80.39% with the highest stability (SD = 4.0%) for task 1. The flow-related dataset has the highest testing accuracy of 77.55% for task 2. Compared to a single-task,

TABLE V
THE PERFORMANCE COMPARISON OF MULTITASKING BY EMPLOYING THE CLASS IMBALANCE AND THE FEATURE SELECTION TECHNIQUES. THE RESULTS OF A FIVE-FOLD CV AND TESTING AVERAGE PERFORMANCE ARE PRESENTED AS MEAN ± SD

Dataset	Cross-validation Performance				Testing Performance			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Flow dataset								
Task 1	82.72±5.19	83.43±4.35	84.70±2.63	84.84±2.10	80.24±4.73	79.27±4.34	83.64±2.68	81.38±3.15
Task 2	78.16±4.06	80.47±4.77	80.89±1.88	82.31±2.47	77.55±4.87	74.06±4.83	80.69±3.95	77.51±3.25
Flag dataset								
Task 1	82.79±3.75	82.63±3.68	84.87±1.91	84.74±1.75	80.39±4.00	78.87±5.94	83.19±1.73	81.50±2.62
Task 2	77.60±5.12	80.97±3.92	81.18±1.32	82.05±3.09	76.90±5.28	74.46±5.41	81.39±4.33	76.75±2.80
Packet dataset								
Task 1	82.17±5.29	82.45±4.75	84.10±2.91	84.78±2.48	80.01±4.50	77.83±4.49	83.47±3.46	79.95±3.55
Task 2	77.02±4.97	80.32±3.65	78.77±1.96	80.94±3.12	77.49±5.75	73.32±3.50	81.27±4.47	76.26±4.38

TABLE VI
THE RESULTS ARE REPORTED BY EMPLOYING THE SELECTED FEATURES IN EACH MODALITY FOR A SINGLE TASK AND MULTITASKING. THE RESULTS OF A FIVE-FOLD CV AND TESTING AVERAGE PERFORMANCE ARE PRESENTED AS MEAN ± SD

Dataset	Cross-validation Performance				Testing Performance			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Flag-related and Packet payload dataset								
Task 1	89.29±3.80	87.83±3.10	92.11±0.91	92.49±4.30	86.08±4.95	84.76±4.95	91.09±4.39	86.08±3.67
Task 2	83.56±4.04	87.11±5.36	84.17±1.73	87.84±2.77	80.69±7.19	81.83±5.00	88.77±2.40	83.64±2.50
Multitask 1	90.23±3.00	91.46±5.00	93.81±2.95	94.51±2.32	88.57±2.91	86.04±5.88	91.32±3.52	90.67±2.23
Multitask 2	85.00±6.00	90.89±4.17	88.92±2.41	90.68±3.26	83.92±6.26	81.73±5.55	90.43±4.07	84.59±2.24
Flow-related and Packet payload dataset								
Task 1	91.25±4.15	90.00±2.68	93.99±1.47	94.41±4.23	88.40±5.01	86.74±5.20	93.14±4.61	88.61±3.67
Task 2	85.74±4.16	89.02±5.67	87.21±1.70	90.65±3.36	83.35±6.65	84.57±4.36	91.09±3.05	86.65±3.07
Multitask 1	92.75±2.98	93.23±5.40	96.79±3.02	97.41±2.32	90.60±3.06	88.86±5.14	93.91±3.56	93.08±2.81
Multitask 2	87.09±5.49	93.43±4.31	91.86±1.82	93.02±3.65	86.98±6.41	84.88±5.89	93.12±3.87	87.17±2.29
Flag-related and Flow-related dataset								
Task 1	92.63±4.08	92.05±2.66	96.14±1.13	96.90±4.34	90.02±5.43	88.80±5.24	94.89±4.40	90.44±3.93
Task 2	88.45±4.24	91.91±4.85	89.89±2.07	92.66±3.11	85.43±7.39	86.91±4.00	92.99±2.44	89.48±3.81
Multitask 1	95.82±3.10	95.38±6.21	99.88±3.53	99.16±2.04	92.95±3.42	91.61±5.28	95.64±3.84	95.64±1.92
Multitask 2	89.10±5.76	95.62±4.63	94.54±2.12	95.74±3.24	89.67±6.24	88.07±5.54	95.71±4.00	89.82±2.25

the multitask model achieved over 2.53% and 2.76% more accuracy in the case of the flag dataset. The flow-related modality improves by 3.06% and 3.78% for tasks 1 and 2 respectively. Similarly, the packet payload dataset improves the testing performance significantly.

Compared to entire features, the flag-related dataset achieves 2.7% and 4.07% better accuracy for tasks 1 and 2, respectively. Moreover, the stability is improved by 1.21% and 0.57%. The flow-related dataset had improved by 4.47% for task 1, while the packet dataset had improved the most by achieving an accuracy of 6.76% for task 2. The experimental results suggest that the feature selection strategy with class imbalanced is highly preferable for attaining maximal CV and testing performance.

C. Performance Evaluation for Different Combinations of Modalities

We conduct experiments by combining the optimum features selected for each modality. According to our findings, combining modalities improves the performance of the proposed model. We use the conjunction of modality in all possible ways for single and multitasked classification. As can be seen in Table VI, the CV and testing performance are significantly improved via this combination.

Single-task: This experiment investigates the impact of feature selection in the training and testing of the proposed multitask model with modality-based feature selection. The experiment is designed to evaluate the proposed multitask model of a single task at a time with selected features using the recursive XGBoost and SULOV methods. The average five-fold CV and testing performance with features chosen are listed in Table VI for tasks 1 and 2 individually across the modalities. In attaining 92.63% and 88.45% CV accuracy, the flag-and-flow modality surpassed the other two combination modalities in both tasks. There is a significant improvement in the proposed model's performance compared to combining the single modality feature selection and the selected features. For example, flag-related only modality achieves a CV and testing accuracies of 80.07% and 76.23% for tasks 1 and 2, respectively. While it combines with the flow-related dataset, it improves the performance by over 12% for both tasks. Combined with the packet payload dataset, it enhances the performance by over 11.18% and 9.51% for tasks 1 and 2.

Furthermore, the testing performance is considerably improved compared to the single modality model with selected

features. The flag-related dataset achieves a testing accuracy of 77.86% and 74.14% for tasks 1 and 2, respectively. In contrast, both combined with the flow-related dataset improves the performance by over 12.16% and 11.29% for both tasks. Combined with the packet payload dataset, it enhances the performance by over 10.54% and 9.21% for tasks 1 and 2.

Multitask: The experiment examines the proposed DL-based model by conducting both the tasks simultaneously with selected features for each modality. The average five-fold CV and testing performance for multitasking across these modalities are presented in Table VI. For tasks 1 and 2, the flag-and-flow dataset shows the highest CV accuracies of 95.82% and 89.1%, respectively.

The flag-and-packet and flow-and-packet datasets show lower accuracies of approximately 2% and 4%, respectively. Compared to the single-task, the multitask model achieves over 3% and 1.5% more accuracy in the case of the flag-and-flow dataset for tasks 1 and 2, respectively. Similarly, throughout the model training phase, the accuracies of the flow-related and packet payload datasets improve substantially. Compared to single modality features selection, combining the selected features enhances the performance. For example, the flow-related dataset achieves 82.72% and 78.16% accuracy for tasks 1 and 2, respectively. However, the combination with the flag dataset shows improvements of 13.1% and 1.58% for the aforementioned tasks. The packet dataset shows the highest improvement by achieving an accuracy of 13.53% for task 1, while the flow-related dataset shows an accuracy improvement of 10.94% for task 2.

In the evaluation phase, the flag-and-flow related modality achieves the best accuracy of 92.95% for task 1. The flag-and-flow dataset shows the highest testing accuracy of 89.67% for task 2. Compared to the single task, the multitask model achieves over 9.65% and 11.18% more accuracy than that in the case of the flag-and-flow dataset. The flow-and-packet combination improves by 12.60% and 8.65% for tasks 1 and 2, respectively. Similarly, the flow and packet payload combination also improves the testing performance by a significant margin.

VII. Conclusion and Future Direction

We propose a novel multitask classification LSTM-based DL approach that utilizes the heterogeneity in the dataset. We employed 18 IoT devices with several malware attacks to formulate a multitasking problem. We took advantage of the potential correlation between various IoT devices and malware by aggregating numerous devices from multiple sources. The data is divided into flow, flags, and packets. The feature selection method is used at the modality level, and the essential features are merged to improve performance. From the experimental analysis, flow-related and flag-related modalities are superior. The combination had the best testing accuracies of 92.63%, 88.45%, and 95.83%, for task 1, task 2, and multitask classification, respectively. Furthermore, it is the most effective in classifying and identifying malware network traffic.

We plan to improve upon our study findings in the future by utilizing additional datasets created from other sources to generalize the proposed multitask model, thereby making the model robust in the malware classification and detection problem. In addition, we will integrate another module or task, such as identifying the device based on the IoT traffic, into the proposed model to strengthen the protection and mitigation procedures. Another future study in the pipeline will involve investigating the proposed model's performance in a real-world setting in the classification and identification of malware and IoT devices simultaneously, particularly for online training.

References

[1] H. N. Saha, A. Mandal, and A. Sinha, "Recent trends in the Internet of Things," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2017, pp. 1–4.

[2] "2020 unit 42 IoT threat report." Unit 42. Mar. 2020. Accessed: Apr. 17, 2022. [Online]. Available: https://start.paloaltonetworks.com/unit-42-iot-threat-report

[3] M. Antonakakis *et al.*, "Understanding the mirai botnet," in *Proc. 26th USENIX Security Symp. (USENIX Security)*, 2017, pp. 1093–1110.

[4] J. Vijayan. "Satori botnet malware now can infect even more IoT devices." 2018. [Online]. Available: https://www.darkreading.com/vulnerabilities-threats/satori-botnet-malware-now-can-infect-even-more-iot-devices

[5] C. Cimpanu *et al.*, "Hajime botnet makes a comeback with massive scan for MikroTik routers." 2018. [Online]. Available: https://www.radware.com/newsevents/mediacoverage/2018/hajime-botnet-makes-a-comeback-with-massive-scan/

[6] L. Pascu. "78% of malware activity in 2018 driven by IoT botnets, NOKIA finds." 2018. [Online]. Available: https://www.bitdefender.com/blog/hotforsecurity/78-malware-activity-2018-driven-iot-botnets-nokia-finds

[7] P.-A. Vervier and Y. Shen, "Before toasters rise up: A view into the emerging IoT threat landscape," in *Proc. Int. Symp. Res. Attacks Intrusions Defenses*, 2018, pp. 556–576.

[8] H. Haddadi, V. Christophides, R. Teixeira, K. Cho, S. Suzuki, and A. Perrig, "Siotome: An edge-ISP collaborative architecture for IoT security," in *Proc. IoTSec*, 2018, pp. 1–4.

[9] T. Zixu, K. S. K. Liyanage, and M. Gurusamy, "Generative adversarial network and auto encoder based anomaly detection in distributed IoT networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–7.

[10] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Security Privacy (ICISSP)*, 2016, pp. 407–414.

[11] R. Mills, A. K. Marnerides, M. Broadbent, and N. Race, "Practical intrusion detection of emerging threats," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 582–600, Mar. 2022.

[12] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. den Hartog, "ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 485–496, Jan. 2022.

[13] I. Ullah and Q. H. Mahmoud, "Network traffic flow based machine learning technique for IoT device identification," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, 2021, pp. 1–8.

[14] Z. Chen *et al.*, "Machine learning-enabled IoT security: Open issues and challenges under advanced persistent threats," *ACM Comput. Surv.*, to be published. [Online]. Available: https://doi.org/10.1145/3530812

[15] M. R. P. Santos, R. M. C. Andrade, D. G. Gomes, and A. C. Callado, "An efficient approach for device identification and traffic classification in IoT ecosystems," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2018, pp. 304–309.

[16] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Managing IoT cyber-security using programmable telemetry and machine learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 60–74, Mar. 2020.

[17] M. Alhanahnah, Q. Lin, Q. Yan, N. Zhang, and Z. Chen, "Efficient signature generation for classifying cross-architecture IoT malware," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, 2018, pp. 1–9.

[18] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 445–458, Jun. 2019.

[19] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1962–1976, Jun. 2021.

[20] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *Proc. ICISSp*, 2017, pp. 253–262.

[21] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019.

[22] R. Zhao. "NSL-KDD." 2022. [Online]. Available: https://dx.doi.org/10.21227/8rpg-qt98

[23] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, 2009, pp. 1–6.

[24] N. Moustafa, 2019, "UNSW_NB15 Dataset," IEEE DataPort. [Online]. Available: https://dx.doi.org/10.21227/8vf7-s525

[25] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Security*, vol. 45, pp. 100–123, Sep. 2014. [Online]. Available: https://doi.org/10.1016/j.cose.2014.05.011

[26] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. B. Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020.

[27] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, 2019, IoT network intrusion dataset," IEEE DataPort. [Online]. Available: https://dx.doi.org/10.21227/q70p-q449

[28] Y. Meidan *et al.*, "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul.–Sep. 2018.

[29] S. Garcia, A. Parmisano, and M. J. Erquiaga, Jan. 2020, "IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic," Zenodo. [Online]. Available: https://www.stratosphereips.org/datasets-iot23

[30] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "BotHunter: Detecting malware infection through IDS-driven dialog correlation," in *Proc. USENIX Security Symp.*, vol. 7, 2007, pp. 1–16.

[31] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet command and control channels in network traffic," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, San Diego, CA, USA, 2008, pp. 1–8. [Online]. Available: https://www.ndss-symposium.org/ndss2008/botsnifferdetectingbotnetcommandandcontrolchannelsinnetworktraffic/

[32] Q. Sun, E. Abdukhamidov, T. Abuhmed, and M. Abuhamad, "Leveraging spectral representations of control flow graphs for efficient analysis of windows malware," in *Proc. ACM Asia Conf. Comput. Commun. Security*, 2022, pp. 1240–1242.

[33] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 646–656, 2013.

[34] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma, "A combination method for android malware detection based on control flow graphs and machine learning algorithms," *IEEE Access*, vol. 7, pp. 21235–21245, 2019.

[35] P. R. Kanna and P. Santhi, "Unified deep learning approach for efficient intrusion detection system using integrated spatial–temporal features," *Knowl. Based Syst.*, vol. 226, Aug. 2021, Art. no. 107132.

[36] A. A. Abd El-Latif, B. Abd-El-Atty, W. Mazurczyk, C. Fung, and S. E. Venegas-Andraca, "Secure data encryption based on quantum walks for 5G Internet of Things scenario," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 118–131, Mar. 2020.

[37] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for Internet of Things in smart city," *Future Gener. Comput. Syst.*, vol. 107, pp. 433–442, Jun. 2020.

[38] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. A. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, Sep. 2019, Art. no. 100059.

[39] Q. Sun, M. Abuhamad, E. Abdukhamidov, E. Chan-Tin, and T. Abuhmed, "MLxPack: Investigating the effects of packers on ML-based Malware detection systems using static and dynamic traits," in *Proc. 1st Workshop Cybersecurity Soc. Sci.*, 2022, pp. 11–18.

[40] J. Singh, D. Thakur, T. Gera, B. Shah, T. Abuhmed, and F. Ali, "Classification and analysis of android malware images using feature fusion technique," *IEEE Access*, vol. 9, pp. 90102–90117, 2021.

[41] S. Lagraa, J. François, A. Lahmadi, M. Miner, C. Hammerschmidt, and R. State, "BotGM: Unsupervised graph mining to detect botnets in traffic flows," in *Proc. 1st Cyber Security Netw. Conf. (CSNet)*, 2017, pp. 1–8.

[42] R. Bhatia, S. Benno, J. Esteban, T. V. Lakshman, and J. Grogan, "Unsupervised machine learning for network-centric anomaly detection in IoT," in *Proc. 3rd ACM CONEXT Workshop Big Data Mach. Learn. Artif. Intell. Data Commun. Netw.*, 2019, pp. 42–48.

[43] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, 2014, pp. 247–252.

[44] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[45] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "IoT malware network traffic classification using visual representation and deep learning," in *Proc. 6th IEEE Conf. Netw. Softw. (NetSoft)*, 2020, pp. 444–449.

[46] R. Shire, S. Shiaeles, K. Bendiab, B. Ghita, and N. Kolokotronis, "Malware squid: A novel IoT malware traffic analysis framework using convolutional neural network and binary visualisation," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Cham, Switzerland: Springer, 2019, pp. 65–76.

[47] I. Baptista, S. Shiaeles, and N. Kolokotronis, "A novel malware detection system based on machine learning and binary visualization," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2019, pp. 1–6.

[48] J. François, C. Wagner, R. State, and T. Engel, "SAFEM: Scalable analysis of flows with entropic measures and SVM," in *Proc. IEEE Netw. Oper. Manag. Symp.*, 2012, pp. 510–513.

[49] S. García, V. Uhlíř, and M. Rehak, "Identifying and modeling botnet C&C behaviors," in *Proc. 1st Int. Workshop Agents CyberSecurity*, 2014, pp. 1–8.

[50] M. Singh, M. Singh, and S. Kaur, "Detecting bot-infected machines using DNS fingerprinting," *Digit. Investig.*, vol. 28, pp. 14–33, Mar. 2019.

[51] M. Dib, S. Torabi, E. Bou-Harb, and C. Assi, "A multi-dimensional deep learning framework for IoT Malware classification and family attribution," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1165–1177, Jun. 2021.

[52] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems," *Sci. Rep.*, vol. 9, no. 1, pp. 1–16, 2019.

[53] T. Abuhmed, S. El-Sappagh, and J. M. Alonso, "Robust hybrid deep learning models for alzheimer's progression detection," *Knowl. Based Syst.*, vol. 213, Feb. 2021, Art. no. 106688.

[54] S. El-Sappagh, T. Abuhmed, S. M. R. Islam, and K. S. Kwak, "Multimodal multitask deep learning model for alzheimer's disease progression detection based on time series data," *Neurocomputing*, vol. 412, pp. 197–215, Oct. 2020.

[55] A. Felkner. "Dataset of legitimate IoT data (VARIoT)." 2022. [Online]. Available: https://www.data.gouv.fr/fr/datasets/dataset-of-legitimate-iot-data/

[56] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Amsterdam, The Netherlands: Elsevier, 2011.

[57] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jan. 2002.

[58] "GitHub—AutoViML/featurewiz: Use advanced feature engineering strategies and select best features from your data set with a single line of code." AutoViML. 2022. [Online]. Available: https://github.com/AutoViML/featurewiz