electronics

MDPI

Article

Exploring Lightweight Deep Learning Solution for Malware Detection in IoT Constraint Environment

Abdur Rehman Khan [1], Amanullah Yasin [2], Syed Muhammad Usman [2], Saddam Hussain [3,*], Shehzad Khalid [4] and Syed Sajid Ullah [5,*]

[1] Department of Computer Science, Faculty of Computing and AI, Air University, Islamabad 44000, Pakistan
[2] Department of Creative Technologies, Faculty of Computing and AI, Air University, Islamabad 44000, Pakistan
[3] School of Digital Science, Universiti Brunei Darussalam, Jalan Tungku Link, Gadong BE1410, Brunei
[4] Department of Computer Engineering, Bahria University, Islamabad 44000, Pakistan
[5] Department of Information and Communication Technology, University of Agder (UiA), N-4898 Grimstad, Norway
* Correspondence: saddamicup1993@gmail.com (S.H.); syed.s.ullah@uia.no (S.S.U.)

Abstract: The present era is facing the industrial revolution. Machine-to-Machine (M2M) communication paradigm is becoming prevalent. Resultantly, the computational capabilities are being embedded in everyday objects called things. When connected to the internet, these things create an Internet of Things (IoT). However, the things are resource-constrained devices that have limited computational power. The connectivity of the things with the internet raises the challenges of the security. The user sensitive information processed by the things is also susceptible to the trusability issues. Therefore, the proliferation of cybersecurity risks and malware threat increases the need for enhanced security integration. This demands augmenting the things with state-of-the-art deep learning models for enhanced detection and protection of the user data. Existingly, the deep learning solutions are overly complex, and often overfitted for the given problem. In this research, our primary objective is to investigate a lightweight deep-learning approach maximizes the accuracy scores with lower computational costs to ensure the applicability of real-time malware monitoring in constrained IoT devices. We used state-of-the-art Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Bi-directional LSTM deep learning algorithm on a vanilla configuration trained on a standard malware dataset. The results of the proposed approach show that the simple deep neural models having single dense layer and a few hundred trainable parameters can eliminate the model overfitting and achieve up to 99.45% accuracy, outperforming the overly complex deep learning models.

Keywords: Internet of Things; deep learning; natural language processing; RNN; LSTM; malware detection

Citation: Khan, A.R.; Yasin, A.; Usman, S.M.; Hussain, S.; Khalid, S.; Ullah, S.S. Exploring Lightweight Deep Learning Solution for Malware Detection in IoT Constraint Environment. *Electronics* **2022**, *11*, 4147. https://doi.org/10.3390/electronics11244147

Academic Editors: Xin Ning, Praveen Kumar Donta and Achyut Shankar

Received: 30 October 2022
Accepted: 25 November 2022
Published: 12 December 2022

1. Introduction

Internet of Things (IoT) is an emerging concept that involves Machine-to-Machine (M2M) communication [1,2]. The prime purpose is to bring the smart capability to everyday life using ubiquitous devices with computational, sensing and communication functionalities, often referred to as IoT devices [3]. With the increasing implications of the IoT devices in various domains such as smart hospitals, homes, streets, offices, etc., we are entering into the revolution of being transformed into smart world [4,5]. However, the implications of IoT devices are challenging mainly because IoT devices are constrained devices, have limited computational capability, and are usually battery-powered [6]. The major mode of communication among the IoT devices is a compact string encoding such as JavaScript Object Notation (JSON) [7]. However, due to computational constraints, communication in IoT devices is prone to cybersecurity threats.

The cybersecurity technologies are designed to provide a shield against computers, networks, programs, and data from attack, mutation, or unauthorized access [8,9]. Among

them, IoT networks have become a significant threat to cybersecurity due to their immense popularity and widespread use [10]. IoT devices enable the industrial evaluation that allows everyday things to become intelligent and ubiquitously assist humans in day-to-day tasks [11]. Billions of people are reportedly taking advantage of IoT platforms, including autonomous vehicle driving. smart healthcare, smart home, etc., to control their environment in real-time [12]. The existing number of connected IoT devices is more than 20.4 billion and is expected to increase exponentially [13].

Recent research has shown that IoT networks are becoming an easy target of cyber criminals [12]. For instance, the rapid spread of users' private information, such as health data, can result in expensive lawsuits [10]. Empirical analysis has shown more than a million confirmed cybersecurity breaches, resulting in the financial loss of approximately $400 million [12]. The Mirai malware infected over 1.2 million IoT peripherals causing the most intense Distributed Denial of Service (DDoS) attacks which further led to the loss of millions due to the downtime [14]. Similarly, ransomware is a mutation of malware that encrypts the user's sensitive information and demands a ransom for decryption, causing up to 200 million dollars in damage [15]. Therefore, urgent attention towards introducing tools and techniques for securing IoT devices information and resources is becoming crucial [10].

Exiting research has critically analyzed the inherent properties of cybersecurity attacks. For instance, an attack signature can be identified from a predefined string, pattern, or rule corresponding to a known attack [16]. Thus, detecting foreseen attacks can be viewed as a pattern recognition problem [15]. The human intellectual capability to recognize patterns is the most primitive cognitive skill that is a foundation for different high-level selection-making in complicated environments [17]. This intellectual capability has resulted in the immense growth of pattern recognition techniques such as image recognition, natural language processing, speech recognition, and so on [18].

Various machine learning-based approaches were introduced in the literature to recognize contextual and visual patterns in different domains [2,19]. Many of these machine learning algorithms are establishing a pivotal role in solving otherwise complicated problems including for instance medical context [20] and cybersecurity issues. Several researchers have produced various datasets based on historical data analysis and security attacks to ease this challenge [21]. Many malware-related text repositories are stored online in textual modality, and various researchers consume such texts for training purposes [22]. However, the enormity and diversity of the texts make it challenging for the researchers to obtain rapid valuable insight. A potential solution is using Natural Language Processing (NLP) to quickly highlight critical information, such as the specific actions and consequences caused by a particular malware [23]. This can assist researchers in understanding the properties of specific malware and ease further advancement.

At present, the algorithms to detect malware in string work on deep learning models. These algorithms work on the probability and statistical theories and therefore are computationally more expensive [16]. The recent advancement in the deep learning models can autonomously learn using billions of trainable parameters, which increases accuracy scores; however, at high computational costs [16]. To increase the accuracy scores, researchers often significantly increase the trainable parameters resulting in high computational costs and complexity [17].

With progressing automation and an increasing number of online systems, various security breaches such as unauthorized access malware attacks, data ransom, and denial of service (DoS) have been reported at an exponential growth rate during the recent years [24]. Nearly 90% of consumers express the concerns on the IoT cybersecurity [25]. Furthermore, this number will probably develop, as per the measurements of AV-TEST organization in Germany [16]. About 26.66 billion devices are currently produced [25] and with the number of IoT devices reaching 75.44 billion by 2025, generating 186 zettabyte of data [26,27]. With IoT systems having billion of devices, zettabyte of data, and sophisticated connections is starting to become an immense security concern [28]. Therefore, it is becoming essential

for organizations to adopt and implement a robust deep learning-based cybersecurity approach to counter cyberattacks [18].

The deep learning models that can autonomously learn using enormous trainable parameters have been shown to mimic near-human-like reasoning. The common inception of deep learning models is extensive computational requirements. However, securing intelligent M2M communication is challenging without integrating deep learning algorithms [29]. Moreover, a research gap exists that investigates the effectiveness of the deep learning models trained on relatively smaller footprints. Therefore, in this research, we are interested in investigating the implication of deep learning models' accuracy that are trained on a smaller footprint. Since information privacy and security is one of the most challenging aspects of IoT, it is becoming critical to enable IoT devices to intelligibly handle security incidents on the runtime [1].

Therefore, in this research, we are interested in finding a lightweight deep learning model that can retrain maximum accuracy with minimum computational costs, which is acceptable in a constrained environment such as IoT. We are using the state-of-the-art vanilla configuration-based RNN [30], LSTM[31], Bi-LSTM [32] model. We trained these models on the widely used MalwareTextDB dataset to test the accuracy scores. We further compared the deep learning-based approaches with the traditional algorithmic machine learning-based approaches to determine their effectiveness. The initial result of the proposed approach shows that lightweight deep learning models can be trained to achieve significant accuracy with fewer computational requirements.

Contributions

Most of malware detection approaches employs relatively complex and computationally expensive deep neural learning models. The approaches that use a simple machine learning algorithms also suffered from low accuracy scores. Therefore, a research gap exists in investigating a lightweight approach that can predict the malware text with acceptable accuracy scores and lower computational costs applicable in IoT environments. The contributions of our research are as follows.

- Exploring a light-weight deep learning-based approach that is applicable in constraint IoT devices.
- Proposing the deep learning models based on minimal parameters and hidden layers.
- Evaluating the proposed approach on the existing state-of-the-art malware dataset.
- Outperforming the existing baselines in terms of accuracy, precision and recall.

The remainder of this paper is organized as follows. In Section 2, we review related work and provide a comparative analysis of the existing state-of-the-art approaches. Section 3 presents our proposed approach and formalize the proposed architecture. Section 4 explains the implementation details including benchmarks, configurations and parameterization process. In Section 5, we critically discuss the results and perform a comparison with the existing approaches. Section 6 finally concludes our discussion and highlights the future research directions.

2. Literature Review

The IoT encapsulates computational, sensing, and communication capability [5]. These IoT devices are interconnected in a networked environment which leads to heterogeneity. Moreover, some of the application areas of IoT, such as smart health care, smart metering, smart homes, etc., are shown to have user trustability issues considering communication security challenges [33]. Therefore, these security issues can not be overlooked, which may compromise the things availability [14].

The first incidence of IoT malware was reported by *Symantec* [34]. This report brought evidence of the malware issue for IoT security. The inherited nature of IoT devices to work in a diverse communication network while providing great connectivity among various devices yet is a hotspot for cyber criminals to infect the IoT devices with crafted malware [35]. A malware-infected IoT device can potentially hijack the massive data of

interest, leading to the consumers' privacy concerns [36]. Furthermore, the malware can traverse up to the IoT platform, injecting the whole IoT service [33]. Therefore, it is vital to investigate AI-enabled IoT devices that can detect malware on runtime with acceptable accuracy scores without impractical computational costs.

Malware is a miscellaneous program that often disguises itself inside the normal data to evade antivirus detection [33]. To counter the inherent disguising mechanism of malware, often pattern matching algorithms are deployed [15]. To detect the abrupt anomaly in the datastream, often Natural Language Processing (NLP) techniques are used [37]. The research has shown that NLP can be effectively deployed in filtering infected pieces of string [14]. The inherent working process of NLP in detecting malware is illustrated in Figure 1. Firstly, the dataset is selected for training the NLP model. This dataset is divided into training and testing datasets to train and test an NLP model, respectively (Figure 1 (Left)). Subsequently, the training dataset is passed to an NLP model, illustrated in Figure 1 (Middle). The present state-of-the-art language model includes Doc2Vec, that are trained on the whole document at once, usually via embeddings. Similarly, Word2Vec is trained on the individual bag of words using transformers. There also exists traditional Latent Semantic Indexing (LSI) to capture the inherent semantics of the data. These models are trained via the training dataset and evaluated using the unseen testing dataset. Finally, a trained model is deployed to classify the clean and infected data shown in Figure 1 (Right). Depending on a particular domain or dataset, a best-performing classifier is selected. These classifiers often use the machine and deep learning algorithms, discussed in the following subsection.

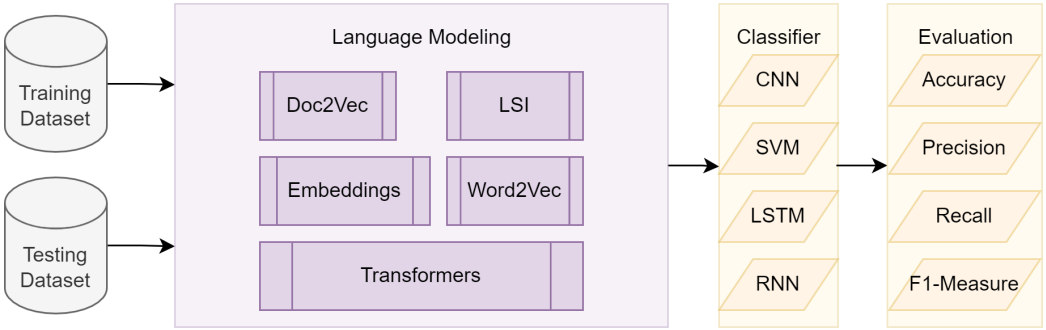

Figure 1. Overall working process of NLP-based malware detection: (**Left**), Datasets (**Middle**), NLP Modeling (**Right**), Classification and evaluation.

Machine learning (ML) is an area of artificial intelligence. The primary purpose of machine learning is to predict, detect and classify data through model's parameters using training dataset [22]. It has recently been shown that ML can offer prospective capabilities to tackle difficult several problems including computer vision, image processing, data mining, and cybersecurity [2,38–40]. Deep learning refers to machine learning technologies utilizing 'deep' artificial neural networks to enhance machine learnability [41]. One of the key features of deep learning is end-to-end training and representation learning, making it a perfect candidate for natural language processing (NLP) [33]. At present, deep learning architectures such as long short-term memory (LSTM), bidirectionally long short-term memory (BiLSTM), gated recurrent unit (GRU), or convolutional neural network (CNN) are often preferred in NLP [42].

The LSTM network architecture consists of three basic gates: input, output, and forget gate [42]. This model can decide which information to keep and what to discard during training. The BiLSTM network, on the other hand, can identify the information relationships from the beginning to the end and vice versa during training [42]. The GRU is similar to the LSTM but reduces the vanishing gradient problem, causing high model saturation during training. The CNN extracts features from the data according to the spatial principle. The Recurrent Neural Network (RNN) is a feed-forward model that enables the networks to capture the temporal dynamics and perform sequential prediction [43].

Most NLP approaches comparatively employ these models at relatively high complexity by adding significant deep hidden layers and trainable parameters [23]. The challenges associated with deep learning include a lack of model internal working and the requirement of extensive training data and computationally powerful resources [44]. In a constrained environment such as IoT, this becomes even more challenging [45].

Roopak et al., used multi-layers essemble of LSTM to detect the DDoS attack in IoT network, achieving a maximum of 98.44% accuracy scores [45]. Researchers in [44] designed an urban-emergency classifier using a CNN deep learning model to detect alarming sounds, such as a fire alarm and subsequently generated textual alert to the user, with 92% accuracy. Taiwo et al. [46], instantiated a CNN model in IoT-enabled cameras, to detect potential robbery attempts, achieving 99.8% accuracy.

However, the existing deep learning approaches are instantiated to enhance the IoT devices functionalities. An investigation on enhancing the security of the data generated by IoT devices using lightweight AI-enabled models is yet to be explored. In the subsequent section, we explore the state-of-the-art technologies used in detecting security issues.

The first baseline method proposed in the classification of malware using Malware-TextDB was by the authors in [22]. They used a generic Support Vector Machine (SVM) and naive bays classifier to identify the malware string. Subsequently, due to the proven success of deep learning-based solutions, the research community started to propose ensemble-based classifiers trained on a deep neural model.

For instance, Villani [47] used word embeddings based on Glove vectors. These embeddings were trained on a general Wikipedia text to represent the tokens. Additionally, they used the LSTM model and subsequently trained a binary classifier using BiLSTM. They used the attention mechanism to fine-tune the weights of the important tokens. Flytxt constructed an ensemble of CRF and NB classifiers [48]. The CRF model was based on lexical and contextual features. They mainly detected whether a token belongs to the tag Action, Entity, or Modifier. The summarized comparative study of the aforementioend approaches is displayed in Table 1.

Table 1. Comparison with the present state-of-the-art approaches.

Year	System	Authors	Features	Model	Accuracy	Limitations
2017	SemEva	Lim et al. [22]	POS	SVM	69.70	Used only statistical method
	SemEva			Naive Bays	59.50	
2018	Villani	Loyola et al. [47]	Wikipedia Tokens	Word-embeddings initialized using Glove vectors	84.47	Trained solely on Wikipedia text data
	Flytxt	Sikdar et al. [48]	NER labels	Ensemble of CRF and NB classifiers	85.28	Used only statistical method
	DM	Ma et al. [49]	NER labels	BiLSTM-CNN-CRF	79.45	Overly complex model configuration
	HCCL	Fu et al. [50]	POS	BiLSTM-CNN-CRF	86.41	
	Digital	Brew et al. [51]	POS, lemma, bigrams	linear SVM classifier	79.45	Used only statistical method
	NLP	Phandi et al. [17]	Tokens	Convolutional neural network with original glove embeddings	76.86	Low accuracy scores
2019	Neural	Ravikiran et al. [52]	NLP labels	Multimodal convolutional neural network	-	Accuracy scores not reported
2020	CRF	Pfeiffer et al. [53]	POS	Dynamic conditional random fields	-	

DM NLP [49] focused on sequence labeling and presented a hybrid approach with BiLSTM-CNN-CRF. They extracted the char-level features using the CNN layer. Using other contextual information, such as POS and labels, as the input to BiLSTM layer trained embeddings and fed the output of the BiLSTM layer into a CRF layer to make entity label prediction. HCCL [50] performed the proposed BiLSTM-CNN-CRF architecture with the significant distinction of using only POS as features. The aim was to build an end-to-end system without excessive dependency on feature engineering or data preprocessing. Ravikiran et al. [52] proposed a three-layer neural network and naive bays classifier

achieving a maximum accuracy score of 65%. Pfeiffer et al. [53] used Conditional Random Fields (CRF) on MalwareTextDB and achieved a maximum accuracy of 86%.

Most of these approaches used relatively complex and computationally expensive deep neural learning models. The approach uses simple machine learning algorithms, which also suffered from low accuracy scores. Therefore, a research gap exists in investigating a lightweight approach that can predict the malware text with acceptable accuracy scores and lower computational costs applicable in IoT environments.

3. Proposed Approach

We proposed the lightweight deep learning approach for detecting malware in the textual strings using natural language processing techniques. Specifically, we are interesting in investigating a deep learning model that uses minimal configurable parameters and hidden layers. To the best of our knowledge, we are the first to determine the effectiveness of the vanilla-configured deep learning models and compare them with the relatively complex configurations. The detailed overview and architectural details of the proposed approach are provided as follows.

3.1. Overview of the Approach

The proposed solution follows traditional malware detection using the NLP mechanism. Mainly, we performed malware detection in four major steps, as shown in Figure 2. Firstly, we shortlisted the database for training our model, illustrated in Figure 2a. This dataset was divided into training, validation and testing datasets, used to train, validate and test the deep learning solution, respectively. Secondly, we performed essential preprocessing steps highlighted in Figure 2b. Precisely, we tokenize each word to facilitate the label extraction task. Based on the tokenized vector, we identified and removed duplicate entries to eliminate bias in the dataset. This also ensures that instead of relying on pattern matching, the model identifies the exact signatures of the malware. We also added a special preprocess handler to label and classify missing or miscellaneous tokens as "UNK". Thirdly, we instantiated our deep learning solution and trained the model based on the batches of the malware text data, shown in Figure 2c. Primarily, we instantiated multiple deep learning models using minimal configuration to determine the most efficient and best-performing solution. The data generator module created batches of the training dataset to minimize computational requirements. Finally, the trained model was evaluated in Figure 2d. Based on the unseen testing dataset, we validated the deep learning models. This process also included further optimization of the deep learning parameters for maximum accuracy scores and classification of the tokens for malware prediction. Complete architectural instantiation details and formalization are provided in the subsequent subsection.

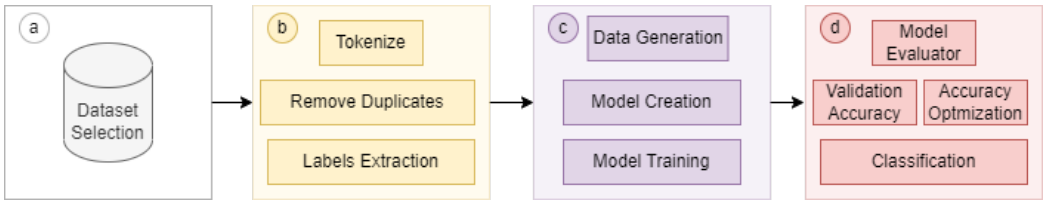

Figure 2. Overview of the proposed approach illustrating the: (**a**), Dataset (**b**), Preprocessing (**c**), Deep learning model (**d**), Evaluation.

3.2. Architectural Design

Existing techniques have utilized either a simple machine learning algorithm or an overly complex deep learning model. In the proposed approach, we are interested in finding a lightweight solution that maximizes the accuracy scores while keeping the computational requirements to a minimum.

This approach utilizes the most recent state-of-the-art deep learning models trained with minimum trainable parameters, thus making it computationally less expensive without a significant drop in accuracy scores. A detailed architecture of the proposed solution is

displayed in Figure 3. Let F be the framework consisting of data, model, and optimization layer, formalized as $F = \{DL, ML, OL\}$.

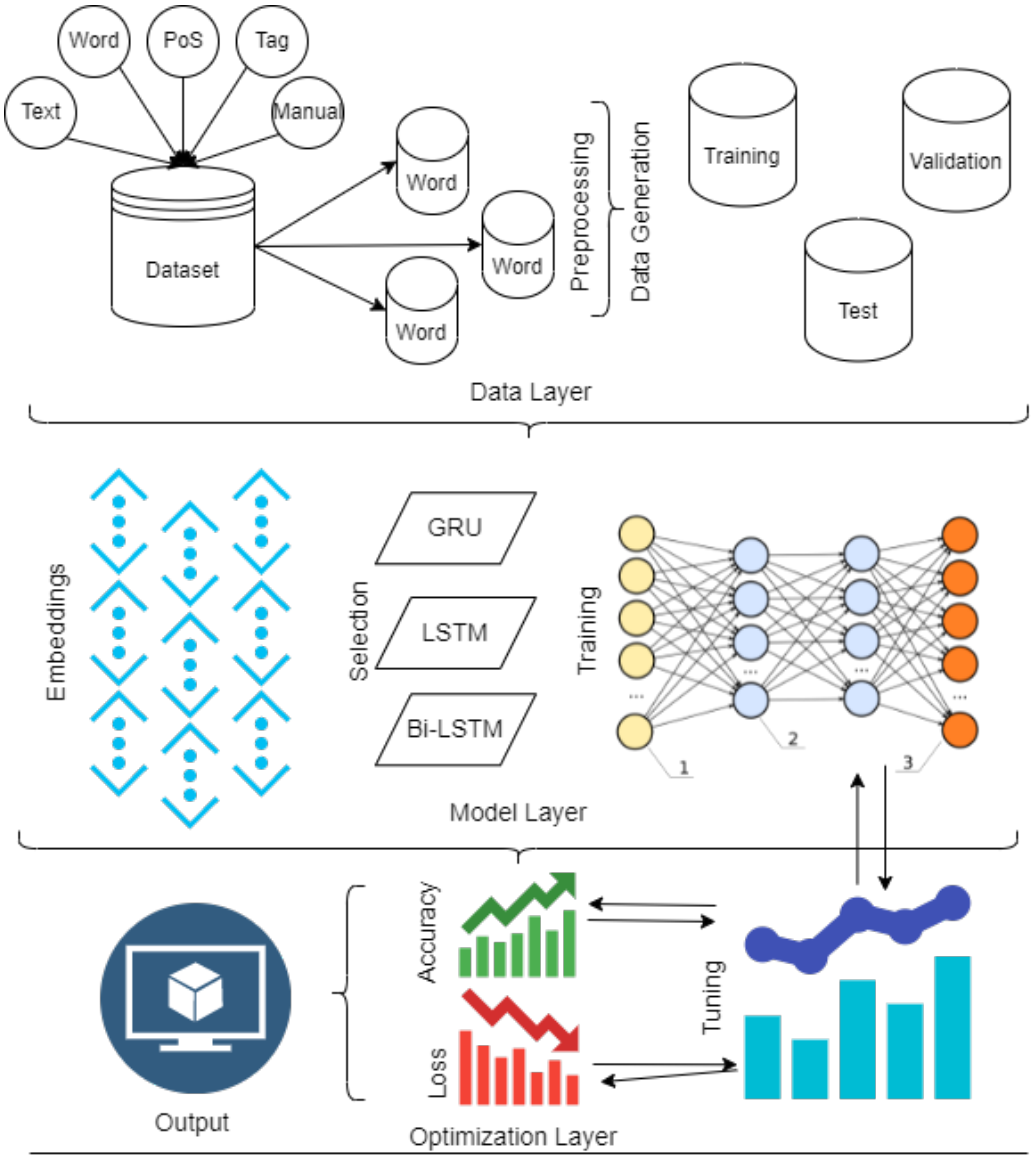

Figure 3. Architectural diagram of the proposed approach illustrating the: (**Top**), Data Layer (**Middle**), Model Layer (**Bottom**), Optimization Layer.

3.2.1. Data Layer

The data layer (DL) encapsulates the dataset M repository. The M consists of a usage manual, test cases, malware sentences, and corresponding annotated labels. Each sentence is divided and annotated into three major parts: (i) Word "w", (ii) Part-of-speech (POS) "p", and (iii) tag "t", e.g., $M = \{w, p, t\}$ where $\|w \wedge p \wedge t\| \notin \emptyset$. The tags are further divided into four categories; Entity "e", Action "a", Modifier "m", and Object "o", e.g., $t = \{e, a, m, o\}$. The e is the subject in focus and initiator of a. The a is referred to as an event such as "registers". The m is the word phrase that links to other words, such as "to". Finally, the o is the recipient of the action, such as "the attacker". We extracted and aggregated all the possible annotated parts present in the dataset. We encoded the M into word-based embedding vector ev_m and target label-based embeddings vector ev_l. We assigned a unique integer number to each unique occurrence of a word in ev_w such that $[\exists w \rightarrow \{0 \ldots N\}]$ where $N \epsilon \|w\|$. Similarly, we mapped the target label ev_l to corresponding ev_w, given as $[\forall i \epsilon w : f(i) \rightarrow t]$. We further split the dataset (DS) into 70% for model training

(DS_t), 15% for validation (DS_v) and 15% for testing (DS_c) the trained model, formalized as $DS = \{DS_t, DS_v, DS_c\}$ where $DS \in M$.

3.2.2. Model Layer

The model layer (ML) transformed each triplet $T(word, POS, tag) \in M$ in DS into word embedding vector ev. The purpose of embedding is to retain semantic relationships for enhanced prediction capabilities in real time. The dimension of each embedding was set to 50, $ev = \{d_1, d_2, d_3, \ldots, d_{50}\}$. The embeddings in ($DS_t$) were fed to the various state-of-the-art deep learning models, including 1:N = {RNN, LSTM, Bi-LSTM}. The (DS_v) embeddings were used to validate the model during training. Finally, the unseen model (DS_c) embeddings were used to obtain actual accuracy scores of the trained model.

The Recurrent Neural Network (RNN) is a deep learning model that consists of four major constructing blocks e.g., $RNN = \{a_i, x_i, y_i\}$, where a_i is the activation function, x is the input batch, and y is the target label at i^{th} state. This primitive building block structure is illustrated in Figure 4. We are using vanilla configuration, which serves threefold. Firstly, this allows minimal computational requirements. Secondly, any input length can be processed, and lastly, model size does not increase with the input size. The RNN model consists of three primary elements; input, activation function, and output. The input size (x) is divided into small batches for fast processing. The activation function (a) is responsible for training the neural network and ensuring correct output (y) predictions.

The RNN activation [54] at each iteration i function is defined as:

$$a^{<i>} = g(W_{aa}a^{i-1} + W_{ax}a^{i} + b_0) \quad (1)$$

where W_{aa} is a weight matrix at the initial neuron and the W_{ax} is a weight matrix at the next neuron. The b is a bias factor to prevent division by zero error.

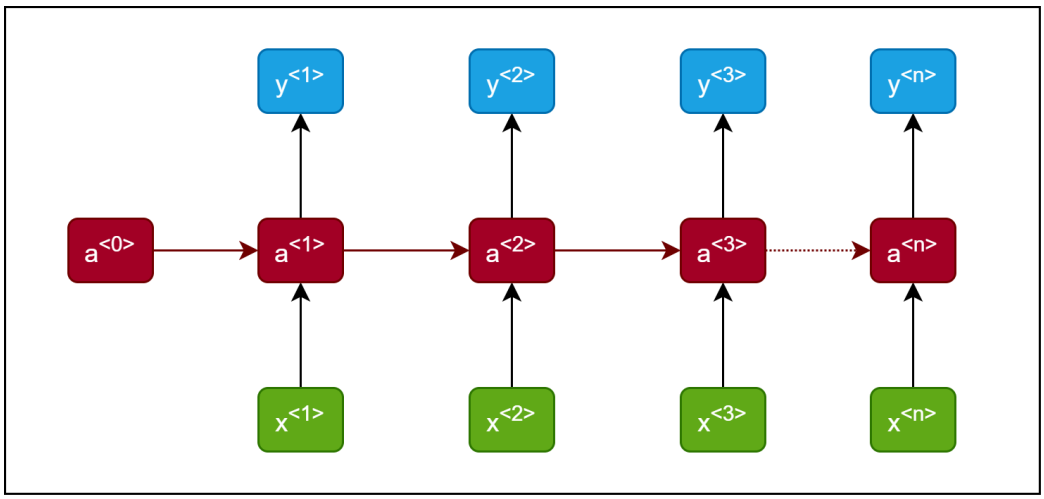

Figure 4. Overview of the RNN architecture [54] illustrating the flow of input (x), output (y), and activation function (a) at each layer.

The Long Short-Term Memory (LSTM) networks are the specialized neural network introduced to eliminate the information loss or gradient decent problem in the RNN. The $LSTM$ is an advanced neural network whose inherent working mechanism extends from the Gated Recurrent Unit (GRU) networks. Therefore, using $LSTM$ serves two major purposes. Firstly, it eliminates the gradient descent problem, and secondly, it provides enhanced performance. Since the $LSTM$ can be trained to discard the unnecessary information from the previous output. The major architecture of $LSTM$ consists of three major constructing blocks e.g., $LSTM = \{f_i^g, X_i^g, y_i^g\}$, where f^g is the forget gate, x^g is the input gate, and y^g is the output date i^{th} state. The primitive building block structure of $LSTM$ is illustrated in Figure 5. We are using vanilla configuration for the same reason

as discussed before. The $LSTM$ model consists of three primary elements; forget, input, and output blocks. The forget block decides which previous information to retain from the previous neuron using the σ function. The input layer is responsible for learning new information using σ and $tanh$ functions. The output layer finally produces the consolidated learning weight (y) predictions. This gated learning process allows maximum learnability and reduces the vanishing of gradients problem. The $LSTM$ activation [54] at each iteration i function is defined as:

$$\sigma = g(W.[y_{i-1}, x_i] + b) Cell_i = tanh(W.[y_{i-1}, x_i] + b) tanh = \frac{e^z + e^{-z}}{e^z - e^{-z}} \quad (2)$$

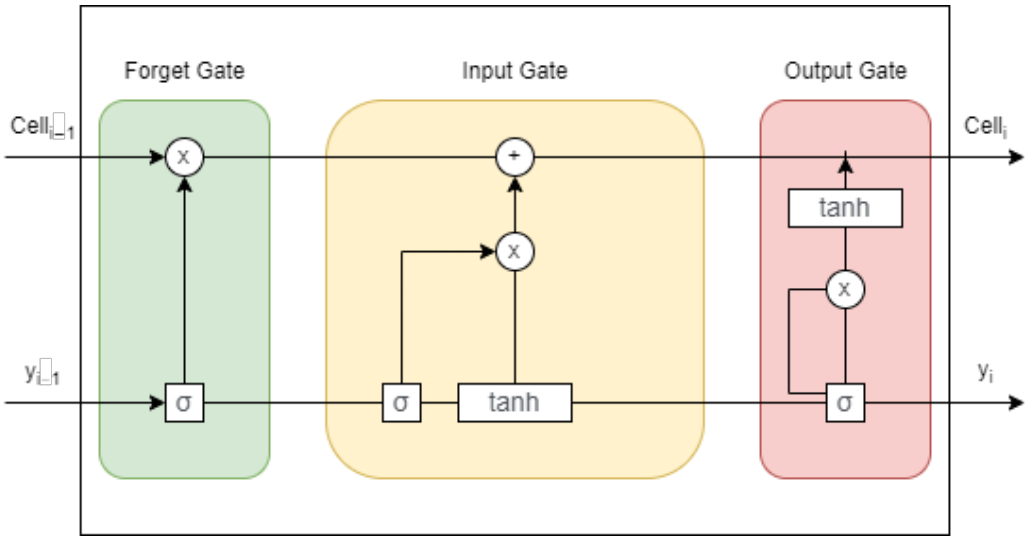

Figure 5. Overview of the LSTM cell architecture [54] illustrating the flow of input (x), output (y), and activation function (α) at the forget, input, and output layer.

The Bi-directional Long-Short Term Memory (Bi-LSTM) networks are the specialized neural network introduced to learn from the future and past input sequences. Hence, $Bi-LSTM$ incorporates double $LSTM$ models. This configuration allows retaining the contextual information, however, at the added costs and complexity. The primitive building block structure of $Bi-LSTM$ is illustrated in Figure 6. We are using vanilla configuration for the same reason as discussed before. The $Bi-LSTM$ model consists of three primary elements; input, LSTM layers, and output blocks. The input sequence is feed-forward to the $LSTM$ layers working in both directions. The output is the aggregated sum of at each iteration i and finally produces the consolidated learning weight (y) predictions.

3.2.3. Optimization Layer

The optimization layer's primary objective is to fine-tune the model during training to achieve the best results. The indicators of the optimized model during training time are accuracy and loss measures. The accuracy denotes the percent of times the model makes a correct prediction, whereas the loss measure shows the amount of information lost during the model's training. The loss function [55] for each model is formalized as:

$$\mathcal{L}(\hat{y}, y) = \sum_{x=i}^{N} \mathcal{L}(\hat{y}^i, y^i) \quad (3)$$

This process is iteratively performed until a convergence point is met in which the achieved accuracy remains the highest while the loss stays the lowest. Based on optimized parameters, the model parameters are finally frozen, and a trained model is generated to

make real-time predictions. The predictions are based on the probabilistic function [56] having k classes:

$$P(y = k|x) = \frac{e^{x^T w_k}}{\sum_{j=1}^{J} e^{x^T w_j}} \quad (4)$$

Finally, the class with the highest probability is selected via the $argmax$ function [56], that returns the target label of the predicted class, formalized as:

$$argmax(y_1, y_2, y_3, \ldots, y_n) = (k_1, k_2, k_3, \ldots, k_n) \quad (5)$$

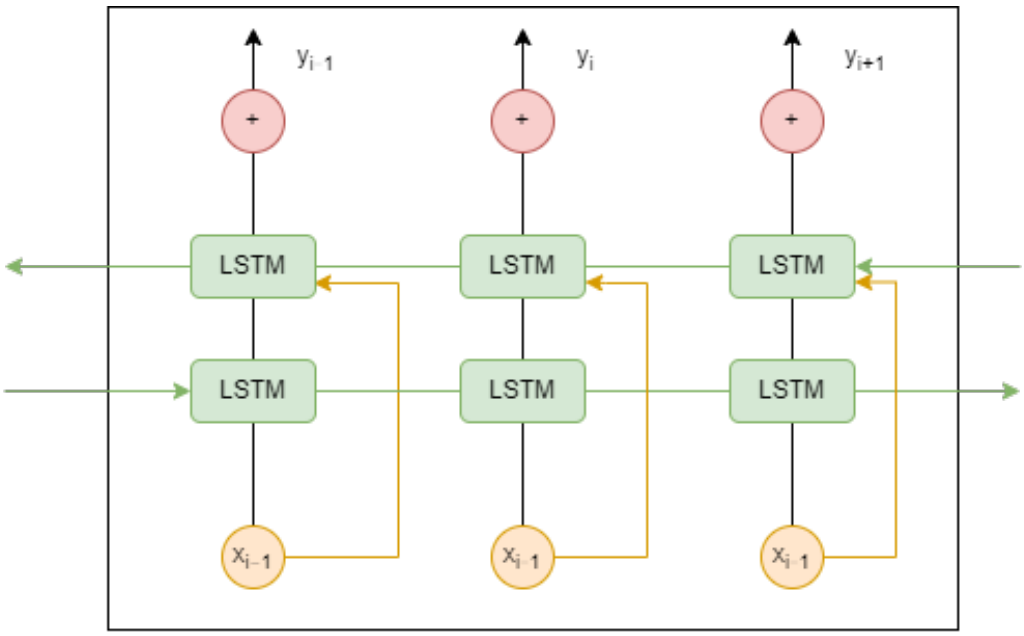

Figure 6. Overview of the Bi-LSTM architecture [55] illustrating the flow of input (x), output (y), and LSTM directions.

4. Implementation Details

The deep learning-based approaches are primarily trained and evaluated on a particular dataset. Based on the model parameterization process, the instantiated deep learning models are trained on the dataset. The parameterization process defines the learning rate, data batches, number of hidden layers, iteration, etc. The detailed discussion on the benchmark, model instantiation, and parameterization process are discussed in the subsequent subsections.

4.1. Benchmark

MalwareTextDB is one of the novels and largest text repositories comprising 26,790 labeled tokens obtained from 2080 sentences [22]. The initial work utilizing this dataset to identify potential malware present in the text has been done on generic machine learning algorithms [17]. Therefore, we used the MalwareTextDB dataset to determine its efficiency compared to the deep learning-based solution. The obtained number of vocabulary, tags, words and a number of instances are displayed in Table 2.

Table 2. Extracted dataset details.

Item	Number	Dataset	Instances
Sentences	6819	Training	4877
Words	168,138	Testing	1045
Tags	7	Validation	1045
Unique words	13,730	Total	6967

4.2. Instantiation

We implemented our approach in Google Colab (https://colab.research.google.com/, accessed on 9 September 2022). The primary programming language used was python 3 (https://www.python.org/downloads/release/python-3101/, accessed on 9 September 2022). To instantiate the model, we used the latest Keras https://keras.io/ and TensorFlow (https://www.tensorflow.org, accessed on 9 September 2022) library. For dataset preprocessing, we used standard libraries such as pandas (https://pandas.pydata.org/, accessed on 9 September 2022) and NumPy (https://numpy.org/, accessed on 9 September 2022). We tested our approach on the three most recent state-of-the-art deep neural architectures, including $LSTM$, $Bi-LSTM$, and RNN. The instantiated architectures are illustrated in Figure 7. Mainly, the input is passed to the model in the form of embeddings. Afterwards, a single layer is used for training the model to minimize the computational complexity, as displayed in Figure 7a for RNN, Figure 7b for $LSTM$ and Figure 7c for $Bi-LSTM$. Finally, a dense layer is added to produce the final prediction. The detailed configuration of the instantiated models are summarized in Table 3.

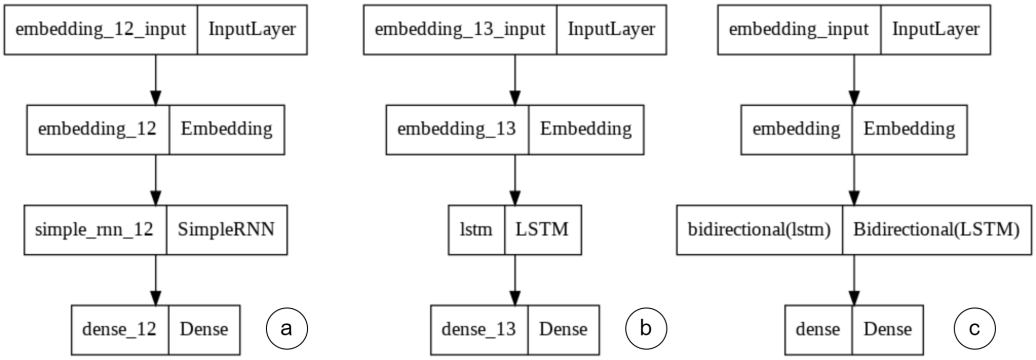

Figure 7. Overview of instantiated models architecture illustrating the (**a**), RNN (**b**), LSTM and (**c**), Bi-LSTM.

Table 3. Instantiated models configuration.

Model	Layer (Type)	Output Shape	Param #
LSTM & RNN	embedding_13 (Embedding)	(None, 141, 32)	439,392
	lstm (LSTM)	(None, 141, 32)	8320
	dense_13 (Dense)	(None, 141, 8)	264
	Total params: 447,976		
	Trainable params: 447,976		
	Trainable params: 447,976		
Bi-LSTM	Layer (type)	Output Shape	Param #
	embedding_13 (Embedding)	(None, 141, 32)	439,392
	lstm (LSTM)	(None, 141, 32)	2080
	dense_13 (Dense)	(None, 141, 8)	264
	Total params: 441,736		
	Trainable params: 441,736		
	Trainable params: 447,976		

4.3. Parameterization

The instantiated parameters are displayed in Table 4 for LSTM, BiLSTM and RNN models. Mainly, we experimented with multiple batch sizes, embedding dimensions,

and loss functions. The epochs were managed by the callback method based on the loss function. We opted only to show the best-performing configurations. The instantiated model parameters are shown in Table 3. Notably, a single hidden layer in the deep learning model was used. The epochs were set based on the last point of accuracy convergence.

Table 4. Instantiated models parameters.

Model Parameters	LSTM	BiLSTM	RNN
Layers	1	1	1
Embedding Size	141	141	141
Dropout Rate	0.2	0.2	0.2
Epochs (Till Convergence)	41	136	79
Learning Rate	0.0001	0.0001	0.0001
Batch Size	128	128	128
Accuracy	0.9945	0.9988	0.9987
Training Loss	0.0154	0.0040	0.0042
Validation Loss	0.1899	0.1977	0.0214
Optimizer	Adam	Adam	Adam
Loss Function		Cross entropy	
Activation Function		Softmax	

5. Results and Discussion

We followed a convention of a 70-15-15 split ratio. Specifically, 70% of the dataset was used to train the models. The 15% dataset was reserved for training and testing purposes. We employed a total of 3 different deep neural network models, including Long-Short Term Memory (LSTM), Bi-Directional LSTM (bi-LSTM), and Recurrent Neural Network (RNN). The results achieved for the best-performing deep neural network are displayed in Table 5. According to our analysis, the RNN achieves the best test accuracy score of 99.46%. The training and validation curves of each model are displayed in Figures 8–10. Similarly, the summary of visualized detailed evaluation measures including accuracy, precision, recall, and F-1 measures, are displayed in Figure 11. The results were obtained through K-fold validation technique with the value of $K = 1$. The obtained results were further averaged by 10 folds and the obtained variance was 2.20.

The results confirm our hypothesis that often vanilla configuration of the less sophisticated deep learning model can provide the best accuracy. Therefore, there exists the need for the researchers to investigate a relatively more uncomplicated configured deep learning model to solve a problem at hand. Moreover, the proposed model configuration requirements are kept to a minimum. Hence, the proposed model has the potential to be used in the IoT devices to monitor and classify the potentially malware infected strings. Therefore, we envision future researchers to provide more innovative deep learning-based solution to the security challenges in domain of IoT.

Table 5. Proposed models evaluation summary report.

Model Parameters	LSTM	BiLSTM	RNN
Test Accuracy	0.9631	0.9681	0.9946
Test Loss	0.2023	0.2274	0.0212

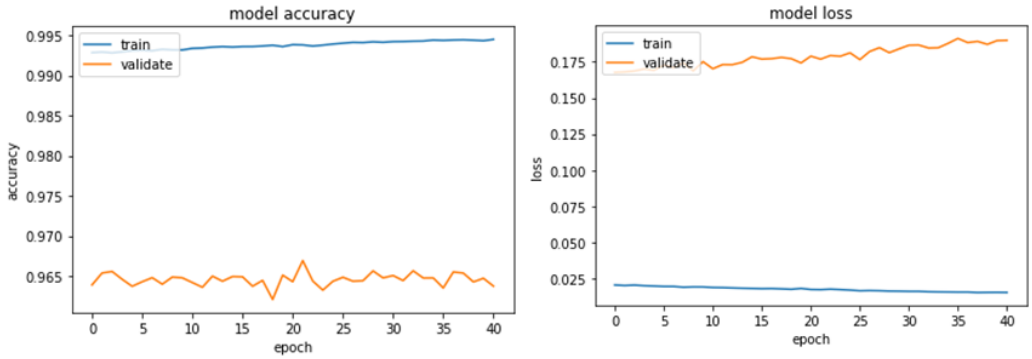

Figure 8. Training and validation loss curves on LSTM model.

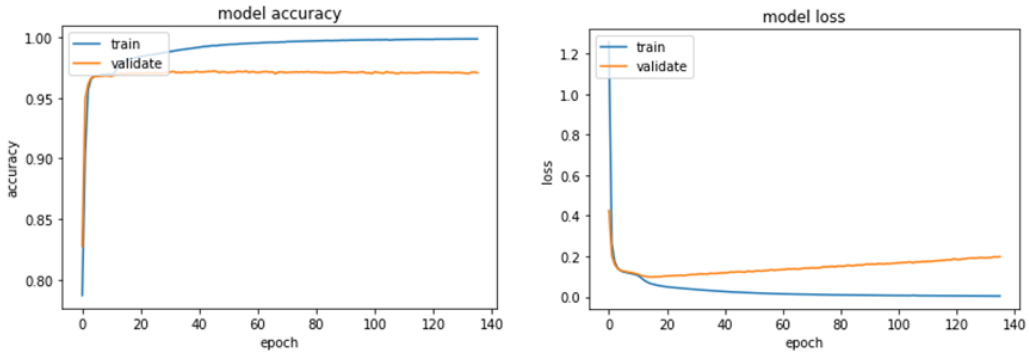

Figure 9. Training and validation loss curves on Bi-LSTM model.

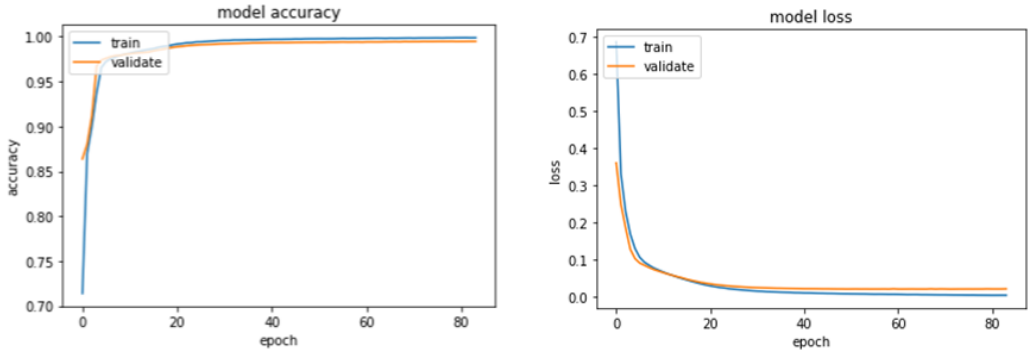

Figure 10. Training and validation loss curves on RNN model.

To compare the proposed approach with the existing state-of-the-art approaches, we employed the MalwareTextDB, which was first released in 2017 by Lim et al. [17]. Initially, they used generic machine learning algorithms such as Naive Bays and Support Vector Machine (SVM). Their achieved accuracy scores were 59.5% and 69.7%, respectively. In 2018, nine teams were invited to the MalwareTextDP competition [17]. The maximum achieved accuracy scores were reported to be 86% based on BiLSTM-CNN-CRF model. We tried experimenting with vanilla LSTM, Bi-LSTM, RNN, and HMM. To the best of our knowledge, a vanilla configuration of the deep learning model for MalwareTextDB classification was not investigated previously. In our detailed analysis concerning statistical methods, we also tested Hidden Markov Model (HMM), which was not previously investigated for MalwareTextDB. After detailed analysis, we found the vanilla RNN deep learning model configuration to give the best results (99% accuracy), surpassing all the previous baseline methods. The statistical method such as HMM also resulted in at most 78% accuracy scores. This comparative analysis is summarized in Table 6. Therefore, the relatively more

straightforward deep learning-based model can be leveraged to surpass computationally expensive statistical algorithms and deep-layered neural networks.

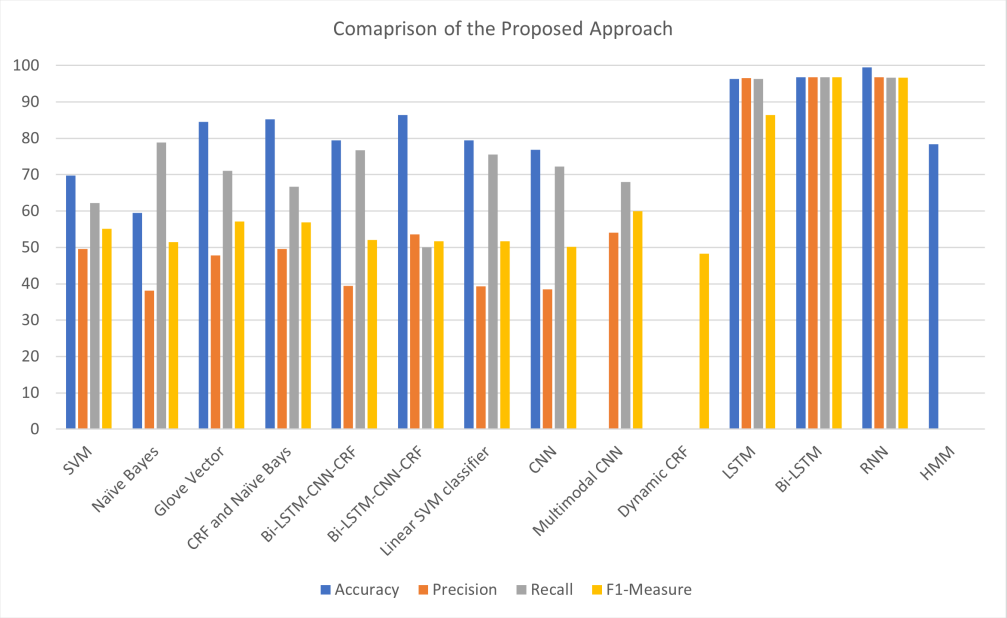

Figure 11. Comparison of the proposed approach with existing state-of-the-art approaches.

Table 6. Comparison with the present state-of-the-art approaches.

Year	System	Authors	Model	Accuracy	Precision	Recall
2017	SemEval	Lim et al.	SVM	69.70	49.55	62.22
			Naive Bays	59.50	38.17	78.89
2018	Villani	Loyola et al.	Word-embeddings initialized using Glove vectors	84.47	47.76	71.11
	Flytxt	Sikdar et al.	Ensemble of CRF and NB classifiers	85.28	49.59	66.67
	DM	Ma et al.	BiLSTM-CNN-CRF	79.45	39.43	76.67
	HCCL	Fu et al.	BiLSTM-CNN-CRF	86.41	53.57	50.00
	Digital	Brew et al.	linear SVM classifier	79.45	39.31	75.56
	NLP	Phandi et al.	Convolutional neural network with original glove embeddings	76.86	38.46	72.22
2019	Neural	Ravikiran et al.	Multimodal convolutional neural network	-	54.00	68.00
2020	CRF	Pfeiffer et al.	Dynamic conditional random fields	-	-	-
2022	Proposed		LSTM	96.31	96.58	96.31
			BiLSTM	96.81	96.78	**96.75**
			RNN	**99.46**	**96.79**	96.63
			HMM	78.33	-	-

6. Conclusions and Future Directions

In this research, we started our discussion on the security issues in the IoT domain. Subsequently, we briefly discussed emerging cybersecurity issues pertaining to malware with potential approaches that provide the solution to increasing security risks. We investigated a lightweight, state-of-the-art deep-learning approach, specifically, using the vanilla configuration of the latest state-of-the-art deep neural models such as RNN, LSTM, and Bi-LSTM, for the prediction of malware text. The instantiated models consisted of single dense layer and only few thousand trianable parameters. To confirm our hypothesis, we used the most extensively available novel MalwareTextDB dataset that could apply to IoT devices based on the standard 70-15-15 datasets splitting method. The proposed approach was thereafter compared with existing baseline methods employing MalwareTextDB pre-

diction approaches which used a complex ensemble of deep learning models. According to our experimental results, we achieved the best accuracy score of up to 99.45% on the simple RNN deep neural model with a single hidden layer, outperforming the existing baselines. Furthermore, we unveiled that a simple statistical model such as Hidden Markov generates a respectable accuracy score of 78%. Therefore, the researchers need to explore a relatively simpler deep learning models before shifting to complex configurations. In the future, we look forward to practically investigating the deep learning models' computation time and storage complexity on IoT devices in real-time using multiple datasets to further determine their feasibility in constrained IoT devices.

Author Contributions: Conceptualization, A.R.K., A.Y. and S.M.U.; methodology, A.R.K.; software, A.R.K., A.Y., S.H., S.M.U. and S.K.; validation, A.R.K. and S.S.U.; formal analysis, A.R.K., A.Y. and S.M.U.; investigation, A.R.K., A.Y., S.S.U. and S.M.U.; resources, A.Y., S.M.U., S.H., S.K. and S.S.U.; data curation, A.R.K.; writing—original draft preparation, A.R.K., S.H. and S.S.U.; writing—review and editing, A.R.K., A.Y., S.M.U., S.H., S.K. and S.S.U.; visualization, A.R.K.; supervision, A.Y. and S.M.U.; project administration, A.Y., S.M.U., S.H., S.K. and S.S.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

References

1. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
2. Naveed, M.; Arif, F.; Usman, S.M.; Anwar, A.; Hadjouni, M.; Elmannai, H.; Hussain, S.; Ullah, S.S.; Umar, F. A Deep Learning-Based Framework for Feature Extraction and Classification of Intrusion Detection in Networks. *Wireless Commun. Mob. Comput.* **2022**, *2022*, 2215852. [CrossRef]
3. Sonar, K.; Upadhyay, H. A survey: DDOS attack on Internet of Things. *Int. J. Eng. Res. Dev.* **2014**, *10*, 58–63.
4. Zohora, F.T.; Khan, M.R.R.; Bhuiyan, M.F.R.; Das, A.K. Enhancing the capabilities of IoT based fog and cloud infrastructures for time sensitive events. In Proceedings of the 2017 International Conference on Electrical Engineering and Computer Science (ICECOS), Palembang, Indonesia, 22–23 August 2017; pp. 224–230.
5. Naveed, M.; Usman, S.M.; Satti, M.I.; Aleshaiker, S.; Anwar, A. Intrusion Detection in Smart IoT Devices for People with Disabilities. In Proceedings of the 2022 IEEE International Smart Cities Conference (ISC2), Paphos, Cyprus, 26–29 September 2022; pp. 1–5.
6. Ko, S.W.; Kim, S.L. Impact of node speed on energy-constrained opportunistic Internet-of-Things with wireless power transfer. *Sensors* **2018**, *18*, 2398. [CrossRef] [PubMed]
7. Rebelo Moreira, J.L.; Ferreira Pires, L.; Van Sinderen, M. Semantic interoperability for the IoT: Analysis of JSON for linked data. In *Enterprise Interoperability: Smart Services and Business Impact of Enterprise Interoperability*; Wiley: Hoboken, NJ, USA, 2018; pp. 163–169.
8. Lu, Y.; Da Xu, L. Internet of Things (IoT) cybersecurity research: A review of current research topics. *IEEE Internet Things J.* **2018**, *6*, 2103–2115. [CrossRef]
9. Alhakami, W.; Alharbi, A.; Bourouis, S.; Alroobaea, R.; Bouguila, N. Network Anomaly Intrusion Detection Using a Nonparametric Bayesian Approach and Feature Selection. *IEEE Access* **2019**, *7*, 52181–52190. [CrossRef]
10. Hassan, W.H. Current research on Internet of Things (IoT) security: A survey. *Comput. Netw.* **2019**, *148*, 283–294.
11. Lee, I. Internet of Things (IoT) cybersecurity: Literature review and IoT cyber risk management. *Future Internet* **2020**, *12*, 157. [CrossRef]
12. Thakur, K.; Hayajneh, T.; Tseng, J. Cyber security in social media: Challenges and the way forward. *IT Prof.* **2019**, *21*, 41–49. [CrossRef]
13. Gopal, T.S.; Meerolla, M.; Jyostna, G.; Reddy Lakshmi Eswari, P.; Magesh, E. Mitigating Mirai Malware Spreading in IoT Environment. In Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 19–22 September 2018; pp. 2226–2230. [CrossRef]
14. Vengatesan, K.; Kumar, A.; Parthibhan, M.; Singhal, A.; Rajesh, R. Analysis of Mirai Botnet Malware Issues and Its Prediction Methods in Internet of Things. In Proceedings of the International Conference on Computer Networks, Big Data and IoT (ICCBI-2018), Madurai, India, 19–20 December 2018; Pandian, A., Senjyu, T., Islam, S.M.S., Wang, H., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 120–126.
15. Humayun, M.; Jhanjhi, N.; Alsayat, A.; Ponnusamy, V. Internet of things and ransomware: Evolution, mitigation and prevention. *Egypt. Inform. J.* **2021**, *22*, 105–117. [CrossRef]

16. Sarker, I.; Kayes, A.S.M.; Badsha, S.; Alqahtani, H.; Watters, P.; Ng, A. Cybersecurity data science: An overview from machine learning perspective. *J. Big Data* **2020**, *7*, 41. [CrossRef]
17. Phandi, P.; Silva, A.; Lu, W. SemEval-2018 task 8: Semantic extraction from CybersecURity REports using natural language processing (SecureNLP). In Proceedings of the 12th International Workshop on Semantic Evaluation, New Orleans, LA, USA, 5–6 June 2018; pp. 697–706.
18. Mahdavifar, S.; Ghorbani, A.A. Application of deep learning to cybersecurity: A survey. *Neurocomputing* **2019**, *347*, 149–176. [CrossRef]
19. Ushmani, A. Machine learning pattern matching. *J. Comput. Sci. Trends Technol.* **2019**, *7*, 4–7.
20. Bourouis, S.; Bouguila, N. Nonparametric learning approach based on infinite flexible mixture model and its application to medical data analysis. *Int. J. Imaging Syst. Technol.* **2021**, *31*, 1989–2002. [CrossRef]
21. Vinayakumar, R.; Soman, K.; Poornachandran, P.; Menon, V.K. A deep-dive on machine learning for cyber security use cases. In *Machine Learning for Computer and Cyber Security*; CRC Press: Boca Raton, FL, USA, 2019; pp. 122–158.
22. Lim, S.K.; Muis, A.O.; Lu, W.; Ong, C.H. Malwaretextdb: A database for annotated malware articles. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1557–1567.
23. Tariq, M.I.; Memon, N.A.; Ahmed, S.; Tayyaba, S.; Mushtaq, M.T.; Mian, N.A.; Imran, M.; Ashraf, M.W. A review of deep learning security and privacy defensive techniques. *Mob. Inf. Syst.* **2020**, *2020*, 6535834. [CrossRef]
24. Shurman, M.M.; Khrais, R.M.; Yateem, A.A. IoT denial-of-service attack detection and prevention using hybrid IDS. In Proceedings of the 2019 International Arab Conference on Information Technology (ACIT), Al Ain, United Arab Emirates, 3–5 December 2019; pp. 252–254.
25. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and security: Challenges and solutions. *Appl. Sci.* **2020**, *10*, 4102. [CrossRef]
26. Zhou, W.; Jia, Y.; Peng, A.; Zhang, Y.; Liu, P. The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. *IEEE Internet Things J.* **2018**, *6*, 1606–1616. [CrossRef]
27. Perwej, Y.; Parwej, F.; Hassan, M.M.M.; Akhtar, N. The internet-of-things (IoT) security: A technological perspective and review. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.* **2019**, *5*, 2456–3307. [CrossRef]
28. Yu, Y.; Li, Y.; Tian, J.; Liu, J. Blockchain-based solutions to security and privacy issues in the internet of things. *IEEE Wirel. Commun.* **2018**, *25*, 12–18. [CrossRef]
29. Syed, N.F.; Baig, Z.; Ibrahim, A.; Valli, C. Denial of service attack detection through machine learning for the IoT. *J. Inf. Telecommun.* **2020**, *4*, 482–503. [CrossRef]
30. Sheikhan, M.; Jadidi, Z.; Farrokhi, A. Intrusion detection using reduced-size RNN based on feature grouping. *Neural Comput. Appl.* **2012**, *21*, 1185–1190. [CrossRef]
31. Althubiti, S.A.; Jones, E.M.; Roy, K. LSTM for anomaly-based network intrusion detection. In Proceedings of the 2018 28th International telecommunication networks and applications conference (ITNAC), Sydney, Australia, 21–23 November 2018; pp. 1–3.
32. Mirza, A.H.; Cosan, S. Computer network intrusion detection using sequential LSTM neural networks autoencoders. In Proceedings of the 2018 26th signal processing and communications applications conference (SIU), Izmir, Turkey, 2–5 May 2018; pp. 1–4.
33. Mimura, M.; Ito, R. Applying NLP techniques to malware detection in a practical environment. *Int. J. Inf. Secur.* **2022**, *21*, 279–291. [CrossRef]
34. Wang, A.; Liang, R.; Liu, X.; Zhang, Y.; Chen, K.; Li, J. An inside look at IoT malware. In Proceedings of the International Conference on Industrial IoT Technologies and Applications, WuHu, China, 25–26 March 2017; pp. 176–186.
35. Wang, H.; Zhang, W.; He, H.; Liu, P.; Luo, D.X.; Liu, Y.; Jiang, J.; Li, Y.; Zhang, X.; Liu, W.; et al. An evolutionary study of IoT malware. *IEEE Internet Things J.* **2021**, *8*, 15422–15440. [CrossRef]
36. Jaramillo, L.E.S. Malware detection and mitigation techniques: Lessons learned from Mirai DDOS attack. *J. Inf. Syst. Eng. Manag.* **2018**, *3*, 19. [CrossRef]
37. Akhtar, M.S.; Feng, T. Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time. *Symmetry* **2022**, *14*, 2308. [CrossRef]
38. Bourouis, S.; Sallay, H.; Bouguila, N. A Competitive Generalized Gamma Mixture Model for Medical Image Diagnosis. *IEEE Access* **2021**, *9*, 13727–13736. [CrossRef]
39. Alharithi, F.S.; Almulihi, A.H.; Bourouis, S.; Alroobaea, R.; Bouguila, N. Discriminative Learning Approach Based on Flexible Mixture Model for Medical Data Categorization and Recognition. *Sensors* **2021**, *21*, 2450. [CrossRef]
40. Almulihi, A.H.; Alharithi, F.S.; Bourouis, S.; Alroobaea, R.; Pawar, Y.; Bouguila, N. Oil Spill Detection in SAR Images Using Online Extended Variational Learning of Dirichlet Process Mixtures of Gamma Distributions. *Remote Sens.* **2021**, *13*, 2991. [CrossRef]
41. Li, H. Deep learning for natural language processing: Advantages and challenges. *Natl. Sci. Rev.* **2017**, *5*, 24–26. [CrossRef]
42. Smagulova, K.; James, A.P. A survey on LSTM memristive neural network architectures and applications. *Eur. Phys. J. Spec. Top.* **2019**, *228*, 2313–2324. [CrossRef]

43. Che, C.; Xiao, C.; Liang, J.; Jin, B.; Zho, J.; Wang, F. An rnn architecture with dynamic temporal matching for personalized predictions of parkinson's disease. In Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, TX, USA, 27–29 April 2017; pp. 198–206.
44. Amudha, S.; Murali, M. Deep learning based energy efficient novel scheduling algorithms for body-fog-cloud in smart hospital. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 7441–7460. [CrossRef]
45. Roopak, M.; Tian, G.Y.; Chambers, J. Deep learning models for cyber security in IoT networks. In Proceedings of the 2019 IEEE 9th annual computing and communication workshop and conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 452–457.
46. Taiwo, O.; Ezugwu, A.E.; Oyelade, O.N.; Almutairi, M.S. Enhanced Intelligent Smart Home Control and Security System Based on Deep Learning Model. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 9307961. [CrossRef]
47. Loyola, P.; Gajananan, K.; Watanabe, Y.; Satoh, F. Villani at SemEval-2018 Task 8: Semantic Extraction from Cybersecurity Reports using Representation Learning. In Proceedings of the 12th International Workshop on Semantic Evaluation, New Orleans, LA, USA, 5–6 June 2018; pp. 885–889.
48. Sikdar, U.K.; Barik, B.; Gambäck, B. Flytxt_NTNU at SemEval-2018 task 8: Identifying and classifying malware text using conditional random fields and Naive Bayes classifiers. In Proceedings of the 12th International Workshop on Semantic Evaluation, New Orleans, LA, USA, 5–6 June 2018; pp. 890–893.
49. Ma, C.; Zheng, H.; Xie, P.; Li, C.; Li, L.; Si, L. DM_NLP at SemEval-2018 Task 8: Neural sequence labeling with linguistic features. In Proceedings of the 12th International Workshop on Semantic Evaluation, New Orleans, LA, USA, 5–6 June 2018; pp. 707–711.
50. Fu, M.; Zhao, X.; Yan, Y. HCCL at SemEval-2018 Task 8: An End-to-End System for Sequence Labeling from Cybersecurity Reports. In Proceedings of the 12th International Workshop on Semantic Evaluation, New Orleans, LA, USA, 5–6 June 2018; pp. 874–877.
51. Brew, C. Digital Operatives at SemEval-2018 Task 8: Using dependency features for malware NLP. In Proceedings of the 12th International Workshop on Semantic Evaluation, New Orleans, LA, USA, 5–6 June 2018; pp. 894–897.
52. Ravikiran, M.; Madgula, K. Fusing Deep Quick Response Code Representations Improves Malware Text Classification. In Proceedings of the ACM Workshop on Crossmodal Learning and Application, Ottawa, ON, Canada, 10 June 2019; pp. 11–18.
53. Pfeiffer, J.; Simpson, E.; Gurevych, I. Low Resource Multi-Task Sequence Tagging–Revisiting Dynamic Conditional Random Fields. *arXiv* **2020**, arXiv:2005.00250.
54. Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]
55. Jeon, J.; Jeong, B.; Baek, S.; Jeong, Y.S. Hybrid Malware Detection Based on Bi-LSTM and SPP-Net for Smart IoT. *IEEE Trans. Ind. Inform.* **2021**, *18*, 4830–4837. [CrossRef]
56. Banerjee, K.; Gupta, R.R.; Vyas, K.; Mishra, B. Exploring alternatives to softmax function. *arXiv* **2020**, arXiv:2011.11538.