Towards an Attention-Based Accurate Intrusion Detection Approach

Arunavo Dey[1], Md. Shohrab Hossain[2], Md. Nazmul Hoq[3], and Suryadipta Majumdar[3(✉)]

[1] Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka, Bangladesh
aronava.d@bubt.edu.bd

[2] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
mshohrabhossain@cse.buet.ac.bd

[3] Concordia Institute for Information Systems Engineering (CIISE), Concordia University, Montreal, Canada
md_oq@encs.concordia.ca, suryadipta.majumdar@concordia.ca

Abstract. With the advancement of 5G and IoT, the volume of network traffic is growing in a tremendous rate (e.g., 235.7 Exabytes (EB) in Internet traffic, a 3.2-fold increase from 2016), leading to an alarming rise in different types of attacks. As a result, the requirements of an intrusion detection system (IDS) are also evolving. In addition to having a large number of flow-based intrusion detection systems powered by machine learning techniques, achieving higher accuracy including higher recall and precision has become equally important. While most of the existing works successfully achieve accuracy, they still strive to achieve a good recall score or minimize the False Negative Rate (FNR) as well as the False Positive Rate (FPR). In this paper, we investigate the potential of combining the state-of-the-art neural network models (i.e., CNN, LSTM, and GRU) with attention mechanisms (where attention helps the model to selectively concentrate on more relevant factors) for improving the accuracy of intrusion detection systems. We evaluate our model with the most recent and state-of-the-art benchmark datasets (e.g., CSE-CIC-IDS-2018, and NSL-KDD) and compare the obtained results with the existing works. Empirical results show that our proposed model outperforms the existing works in terms of accuracy while achieving a higher recall score (e.g., a maximum recall of 100%, 99.91% for CSE-CIC-IDS-2018, and NSL-KDD datasets, respectively) and higher F1-Score (e.g., a maximum F-1 score of 100%, 99.22% for CSE-CIC-IDS-2018, and NSL-KDD datasets, respectively).

Keywords: Network security · Intrusion detection · Attention · Neural network

X. Yuan et al. (Eds.): QShine 2021, LNICST 402, pp. 261–279, 2021.
https://doi.org/10.1007/978-3-030-91424-0_16

1 Introduction

In the modern era, network traffic is a growing entity, which is only getting bigger with each new addition of technologies (e.g., IoT, 5G). According to the global reports by CISCO, Internet traffic will rise to 7.7 Exabytes per day and 235.7 Exabytes (EB) per month in 2021 [1]. In addition, the growing traffic due to 5G and IoT is adding huge volume to the already existing load in the network (e.g., estimated to rise by three-fold within 2025 [2]). With this increase in the network traffic and data, the number of security breaches and total records exposed per breach continue to grow as there was 776% growth in network attacks from 2018 to 2019 [3]. To detect such security breaches, the intrusion detection is gaining a fair share of its importance and on the way to gain more.

Being a widely-studied topics of cybersecurity, intrusion detection has been experimented with a lot of techniques, including neural network based approaches (e.g., [4–9]); where all of those recent works show the effectiveness (in terms of accuracy) of the detection approach. However, most of them are not focused on improving recall score, False Negative Rate (FNR), and False Positive Rate (FPR), which are also equally important for an intrusion detection system. Moreover, most of those works are not based on the most recent benchmark datasets (e.g., NSL-KDD [10], CSE-CIC-IDS-2018 [11]). Even though LuNet [12] shows comparatively better accuracy and recall score for the NSL-KDD dataset, it is still burdened with a large architecture (later we will compare our proposed method with LuNet). Moreover, apart from achieving accuracy, what matters most for an intrusion detection system is achieving a low false-positive rate (FPR) and low false-negative rate (FNR) by achieving high recall and precision values [13,14]. As recall and FNR are related, we can get the value of FNR simply by calculating recall. While accuracy is a good metric for a balanced datasets that contain almost the same amount of false positive and false negative data, F-score is better suited for unbalanced datasets like intrusion detection data. Most of the existing works focused on the accuracy measure more than other metrics i.e., recall, F-Score, etc. None of the existing works achieves a good recall score nor evaluates their performance on all the state-of-the-art benchmark datasets (including the latest CSE-CIC-IDS-2018 dataset [11]). Additionally, most of the existing models are comprised of several layers which renders computational overhead.

To overcome this limitation, in this paper, we investigate the potential to leverage the advantages of both Convolutional Neural Network (CNN) for extracting features from input and Recurrent Neural Network (RNN) for extracting sequential features together with the advantage of attention mechanism for better performance in sequential features (which is mostly used in NLP tasks including similar problem like ours). This proposed method is evaluated with both NSL-KDD [10] and CSE-CIC-IDS-2018 datasets [11]. Our obtained results show that our proposed model outperforms all the state-of-the-art models in terms of F1-Score as well as recall score. Particularly, our results indicate that our model can achieve a higher F1-Score with the highest recall score amongst all of the existing neural network models used in intrusion detection with a

simpleton architecture that is faster to train and can perform on raw data, excluding any need for scaling or normalization. Thus, this work can help to detect almost all the latest variations of Web, DDos, Botnet, HeartBleed, and Infiltration attacks (exploited in the CSE-CIC-IDS-2018 datasets [11]) while achieving a higher F1-score, and hence shows its potential for the networks where recall score is given priority over accuracy.

The main contributions of this paper are as follows.

- As per our knowledge, we are the first to investigate the potential of combining the CNN, LSTM, and GRU models with attention mechanisms for improving different aspects of accuracy metrics (including recall, F1-score, etc.) of intrusion detection systems.
- We utilize state-of-the-art benchmark network traffic datasets (including NSL-KDD [10] and CSE-CIC-IDS-2018 [11] comprising various network attacks to evaluate our proposed model.
- Through experimental results, we demonstrate that our proposed model can improve the accuracy (including F1-Score and recall Score) in comparison to the existing neural network-based models (e.g., an average F1-Score value of 95.41% by our model vs. 81.87% for CNN and 83.50% for CNN-LSTM using CSE-CIC-IDS-2018 [11] dataset).

The rest of the paper is organized as follows. Section 2 reviews existing works. In Sect. 3, the proposed approach is presented. Section 4 describes the dataset as well as experimental setup and model details. Section 5 presents our experimental results with the comparison of the performance of the proposed model with existing works. Finally, Sect. 6 provides concluding remarks including future directions.

2 Related Work

There are several works (e.g., [4–9]) applying deep neural networks for intrusion detection, where different deep learning mechanisms, e.g., CNN, RNN, and LSTM, are leveraged. Recently, attention-based approaches [15–17] are also being popular in intrusion detection systems. In the following, we further discuss the related works and analysis the current gaps in the intrusion detection system.

2.1 Neural Network Based Intrusion Detection Systems

Neural Network (NN) and Deep Neural Network (DNN) have been used in intrusion detection for a long time. Botros et al. [8] study the methods and apparatus for training a neural network model to apply for intrusion detection. They outline a model workflow for leveraging neural network algorithms in intrusion detection without evaluating the model with any particular dataset. Shun et al. [18] study the application of neural network on DARPA dataset [19] with the simplest configuration consisting of input and output layers coupled with a few hidden layers

and demonstrate the potential of neural networks in this domain without applying their model on various datasets or specifying other evaluation details such as recall and FPR. Chiba et al. [20] propose a neural network with back propagation for intrusion detection and exhibit the neural network performance on the KDD-99 Cup dataset [21] by achieving a lower FPR and higher recall score than the others. Mahalingam et al. [22] propose an intelligent intrusion detection system based on the combination of signature analysis and anomaly analysis using a neural network. However, this model lags behind others in terms of accuracy, which provides only 70% accurate results. Su et al. [23] use an improved version of the rough set-particle swarm optimization algorithm for improving intrusion detection systems where they cluster similar data and reduce the difficulty in the identification of data. They use an open dataset with a simple SVM-based approach and show that their proposed model could achieve a 98% recall score.

Recently, different variations of DNN, such as CNN, LSTM, RNN, etc., and different combinations among them are being used in detecting intrusion. The most recent work with CNN on IDS has been demonstrated by Mendon et al. [24]. They use a Tri-CNN model with a soft-sign activation function to detect intrusion and get a good accuracy of 98% while reducing the training time significantly by 36%. However, they use the CSE-CIC-IDS-2017 dataset [25] instead of the most recent one. Kim et al. [26] use a CNN-based approach for intrusion detection focusing on only DoS attacks. They use the KDD-99 and CSE-CIC-IDS-2018 datasets to evaluate the performance and show that their CNN-based approach is better than any RNN-based approach. Nonetheless, as mentioned earlier, this works is only focused on a single attack. Whereas Wang et al. [27] develop a CNN-based approach and show the performance only on the NSL-KDD dataset. Bandyopadhyay et al. [28] recently propose an optimized deep CNN model with evaluation results based on experiments only on the KDD-99 dataset, but many of the recent works mentioned above already outperform their work (84% accuracy) in terms of accuracy and other evaluation metrics.

A combination of CNN with LSTM model (CNN-LSTM) is also popular in IDS. Sun et al. [29] apply a category weight optimization method on a CNN-LSTM based method to detect intrusion. They use the CSE-CIC-IDS-2017 dataset instead of the most recent one and show that the overall accuracy is 98.67% and F1-Score is 93.32%. However, for specific attack types (i.e., Heartbleed and SSHPatator attacks), they obtain a low detection accuracy. Kim et al. [30] use LSTM cells with recurrent neural network for intrusion detection though not surpassing the higher accuracy and recall achieved by the previous works. Kuang et al. [31] propose an one dimensional (1-D) CNN-LSTM model and show its accuracy on CSIC-2010 dataset [32]. Where Hsu et al. [9] propose almost similar CNN-LSTM model and show its effectiveness on NSL-KDD dataset. Hsu et al. [33] show the performance of the LSTM model and the combination of CNN with the LSTM model (CNN-LSTM). Using the NSL-KDD dataset, they show that both the proposed methods achieve a better accuracy than the RNN based models.

2.2 Attention and Encoder-Decoder Based IDS

In recent times, attention mechanism incorporated with neural networks have become widely available. Most of them use the auto encoder-decoder structure or neural machine translation approach. In the following, we will elaborate on the works regarding attention based model.

Several recent works incorporated attention-based mechanisms in neural networks. For example, Liu et al. [15] use an attention-based bi-directional GRU model for IDS and show the model performance on UNSW-NB-15 [34], NSL-KDD [10], and KDD-99 [21] datasets instead of using the most recent dataset. Liu et al. [17] use the payload information for web attack detection with their attention based neural network but additional payload analysis before classification adds up to already existing workload occupying much time. Although they achieve a higher accuracy and lower FPR, they evaluate their model performance only on CSIC-2010 dataset instead of the most recent one.

Moreover, the latest encoder decoder based approaches also show promising performance. Basati et al. [35] propose an IDS named APAE specifically for IoT networks where they use two parallel encoder-decoder. They measure the efficiency of the system using UNSW-NB-15, KDD-99, and CSE-CIC-IDS-2017 datasets, instead of the most recent dataset. Sekhar et al. [36] also use a deep autoencoder with fruitfly-Optimization-based IDS and measure the performance on UNSW-NB-15 and NSL-KDD datasets instead of using the most recent dataset. Nathan et al. [37] propose a non-symmetric autoencoder based intrusion detection system and evaluate the performance using the KDD-99 Cup and NSL-KDD datasets showing accuracy's for each of the attacks separately. Tang et al. [38] use an encoder-decoder network for detecting zero day attack by applying neural machine translation converting http requests through encoder and converting them back using decoder and detecting attacks based on the similarity (BLEU score) between original and translations. However, instead of showing their performance on any benchmark dataset they evaluated their performance on a simulated environment. Moradi et al. [39] use stacked autoencoder for detecting web attacks on the CSIC-2010 dataset though not surpassing the previous achieved performance. Mac et al. [40] follow an unsupervised approach and study the performance of autoencoder on CSIC-2010 dataset detecting web attacks; ending up with regularized autoencoder outperforming eleven other encoder-decoder with an AUC score of 98.23% and recall score of 94.62%. Yan et al. [16] use neural machine translation with NLP techniques for web attack detection on CSIC-2010 dataset. However, they perform their technique separately on the dataset for each type of the attack individually achieving a highest recall of 98.29% and lowest recall of 43.98%.

2.3 Gap Analysis

Most of the models achieve better accuracy without achieving better recall scores and F1-Scores. Moreover, the better accuracy and better recall score achieving model LuNet [12] involve a batch normalization between model layers with a

huge architecture. Thus, achieving better performance with a light-weight model is what has driven us, and nevertheless, our model surpasses them all on the benchmark datasets by detection rate achieving higher recall and higher F1-score from almost all of them, and its performance is also elaborated on the latest dataset (i.e., CSE-CIC-IDS-2018 [11] and NSL-KDD [10]).

Unlike others, in this paper, we follow a straightforward approach by combining the NLP aspect with CNN and LSTM networks by enriching them with attention mechanism. Instead of dealing with each attack separately, we generalize all of the attacks in an anomaly class and perform a binary classification.

3 Proposed Approach

Generally, attention-based models regarding IDS, use autoencoder-decoder or stacked layers that make the model heavy and computationally expensive. To keep our model simple, we combine CNN and LSTM models together with the attention mechanism. As LSTM in spite of being proven worthy, fails to capture long-term dependencies, thus we combine GRU with the above-mentioned combination and further combine an attention layer to focus on the important parts. The detailed structure of our model is shown in Fig. 1, which shows that the model is constructed with an embedding layer following an input layer, followed by a single 1D convolution layer (best suited for models dealing with textual classifications), and as usual followed by a max-pooling layer. We combine the LSTM with GRU as well as with an attention layer, later concatenating those two-fed layers before the final output layer. A detailed description of these layers is given below.

3.1 Input Pre-processing

Tokenization. We treat the HTTP requests as text sequences. If the request is in the format:

$$X : x_1, x_2, ..., x_n$$

containing n columns, where the n columns are converted as string entries and concatenated and then tokens are generated according to punctuation marks. Tokens are the smallest part of a string or sentence. A token could be a word, number, or punctuation. In the tokenization process, the words are converted into integer numbers so that it would be efficient to feed them into the neural network. Below example, shows a sample sentence with its tokens converted to an integer sequence: Sample Sentence: "*Hello, this one sentence!*" and its corresponding integer Sequence: `125`, `778`, `3`, `63`.

Vocabulary. All the sentences are used to generate the vocabulary, and the tokens exceeding a certain threshold are being selected. The vocabulary consists of all unique tokens in the datasets. This vocabulary is also a mapping of the words and integer numbers. It has been used in encoding the sentences into number sequences and also does the reverse.

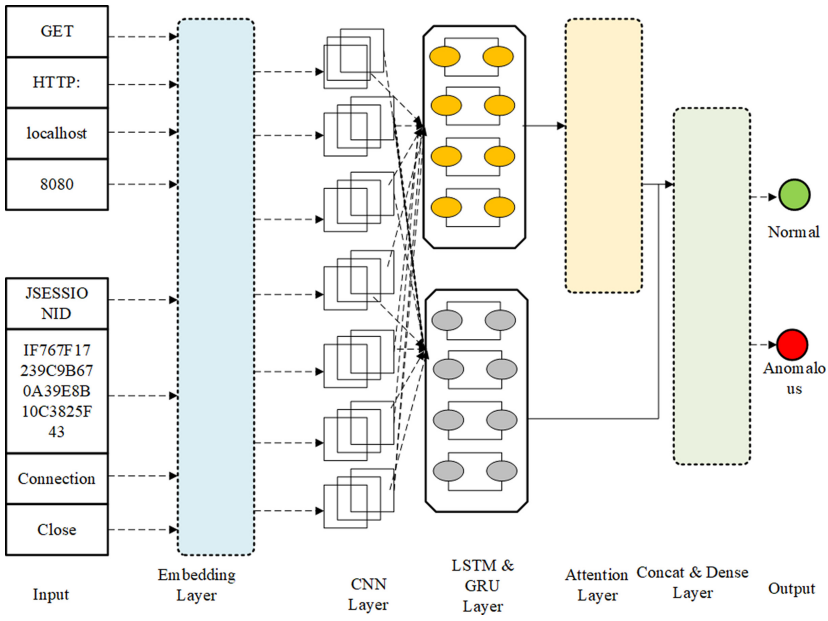

Fig. 1. Structure of our model

Padding and Truncating the Sequence. After the sequences are tokenized, formed into number sequences, they are padded and fed to the embedding layer. Padding is just adding zero, either the beginning or end of the integer or number sequence. The padding ensures that all the sentence lengths are equal. Sometimes instead of padding, truncating is also used. Truncating means removing some tokens from the beginning or end to make all the sentence lengths equal.

Embedding. After processing, these sequences are fed into the embedding layer, where these sequences are formed as embedded vectors. Embedding layers vectors holds the relationship among the words. Also, the corpus of the embedded vectors is previously built on the vocabulary, and the dimension is fixed after a number of observations.

3.2 Input Layer

The input layer receives an embedded vector array of fixed length and simply passes this to the convolution layer after inspecting the specified input dimensions. The input layer consists of the pre-processed data, which has been described in the previous section.

3.3 CNN Layer

The Convolution layer performs the convolution operation on the input fed from the input layer. If the convolution operation is performed upon m word vectors with weight matrix $W \in R^{l*m}$ then

$$O_i = f(X_{i+m-1} * W_i + b_i).$$

This layers works to find the features from the input sequences by operating with several filters. After completing convolution operation, max pooling is performed upon the output:

$$M_i = maxO_i.$$

This max pooling contributes to finding more important features from the collected ones.

3.4 LSTM Layer

LSTM is one of the stronger variants of RNN, capturing long-term dependencies successfully. Here we use the context learned from the CNN as input to this layer to generate the final feature map:

$$h_i = lstm(M_i).$$

As we are dealing with inputs as sequences and some features may remain co-related sequentially; we use LSTM to capture these specifically. For example, some specific sources may be the sources of repeated attacks, and extracting sequential information may be helpful. LSTM cell calculates and saves the information it finds relevant in cell state and saves the previous information in hidden states. We pass the outputs and hidden state to the next layer to calculate the attention score.

3.5 Attention Layer

We feed the output of the maxpooling layer to our attention layer through a LSTM layer. The LSTM layer gives us the outputs as well as forward (hidden) states and back (cell) states. We feed the forward states along with outputs to our own implemented Bahdanau Attention layer. Bahdanau Attention layer calculates the score as

$$score(F, O) = {v_a}^T tanh(W_1 F + W_2 O).$$

where F and O are the forward states and output of the previous layer, respectively, and the C is the context vector or Output of attention layer and W is the attention states or attention weights.

According to the Bahdanau [41] Attention layer, we compute the attention weights and context vector using the following formula:

$$\alpha_{ts} = \frac{exp(score(F, O_s))}{\sum_{i=1}^{S} exp(score(F, O_i))},$$

$$c_t = \sum \alpha_{ts} O_s.$$

The attention mechanism focuses on the more important parts by calculating scores from the forward states and outputs and generating context vectors from them using attention weights. These context vectors work as the summarized output, which is being fed to the Concatenation Layer.

3.6 GRU Layer

The output from Maxpooling layer is also fed to a GRU layer. GRU is the variation of RNN which uses update gate and reset gate to calculate the output

$$z_i = \sigma(WM_i + Uh_{t-1}),$$
$$r_i = \sigma(WM_i + Uh_{t-1}),$$
$$h_i = tanh(WM_i + r_i \odot Uh_{t-1}).$$

GRU is another variant of RNN which is good at extracting sequential information. We leverage this to keep the important features that may be missed by the attention layer and used afterward together with attention layers output.

3.7 Concatenation Layer

The concatenation layer concatenates the outputs from the attention layer and GRU layer to generates output for the next layer. This layer takes inputs from the attention layer and from the GRU layer which should have the same dimensions except for concatenation axes. The concatenated output then feed to the dense layer. If W_1 and W_2 be the wight matrices and x and y are the feature matrices, then the contention could be expressed as:

$$W[x, y] = W_1 x + W_2 y.$$

3.8 Dense Layer

Lastly, the output layer performs sentiment classification where we use sigmoid activation function the mostly used in the neural network. The sigmoid activation function converts the matrix of numeric values into non-linear binary values of 0 and 1. As the loss function, Cross-Entropy has been used.

4 Experimental Setup

We build our model using the TensorFlow [42] backend, Keras [43], and scikit learn packages [44], and we conduct the training on a personal computer (pc) with Intel(R) Xeon(R) Haswell CPU @ 2.30 GHz processor and 12.0 GB RAM with a T4 GPU. For comparison, we also implement other state-of-the-art machine learning algorithms. We use the Nadam optimizer [45] to adjust weights and the categorical cross-entropy as the cost function during the training. In addition, the configuration of the hyper-parameter that is being used is described in Table 1.

Table 1. Hyper-parameter configuration

Hyper parameter	Value
CNN layers	1
CNN filters	256
LSTM layer	1
LSTM nodes	20
GRU layer	1
GRU nodes	20

4.1 Datasets

Performance evaluation of a neural network design is highly dependent on the dataset used. Many datasets collected for IDS contain a significant amount of redundant data [46,47], making the experiment questionable where on the other hand using the latest available dataset proves the models performance in dealing with the latest types of attacks. To ensure the effectiveness of the evaluation, we perform our experiment on four different datasets: CSE-CIC-IDS-2018 [11] (which is one of the most recent benchmarking datasets), NSL-KDD [10], Original KDD-99 Cup [21], and CSIC-2010 [32].

Table 2. Attack distribution in KDD-99 [21] and NSL-KDD datasets [10]

Attack types	KDD-99	NSL-KDD
DoS	3883370	45927
Probe	41102	11656
R2L	1126	995
U2R	52	52
Normal	972781	67343

Among these, KDD-99 Cup [21] is the oldest one used for benchmarking. After that the NSL-KDD [10] dataset was developed and since then it has been utilized in almost every work as a benchmark for the intrusion detection system. The NSL-KDD [10] dataset is an updated version of the KDD-99 Cup dataset which has some drawbacks [48] that can affect the accuracy of the model. In addition, unlike the original KDD-99, NSL-KDD doesn't contain the redundancy in the training set, and neither has duplicate records in the test set. This dataset contains 41 features and an additional feature as the label for each record. In this paper, as we concentrate our model on binary classification, We divide the data into two categories normal and anomalous data. Table 2 provides attack distribution that has been used in the KDD-99 and NSL-KDD datasets.

The CSIC 2010 dataset is another widely used dataset, which is divided into two subsets. The first subset contains 36,000 normal requests, and the second

subset contains 36,000 normal and more than 25,000 abnormal requests. Basically, this data set is used to train unsupervised machine learning models where the first set is used for training. The second subset is used to measure the models accuracy in detecting the unknown attacks. We use it to compare with how the model performs on this data set if used in a supervised way.

Table 3. Attack types and durations in CSE-CIC-IDS-2018 datasets [11]

Attack types	Tools	Attack duration
Bruteforce	FTP – Patator, SSH – Patator	One day
DoS	Heartleech	One day
DoS	Slowloris, Slowhttptest, and others.	One day
Web	DVWA, XSS, and others	Two days
Infiltration	Nmap and portscan	Two days
Botnet	Keylogging and others	One day
DDoS+PortScan	LOIC	Two days

Canadian Institute for Cybersecurity (CIC) generated IDS datasets in 2012, 2017, and 2018 among which CSE-CIC-IDS-2018 [11] is the most up-to-date dataset for IDS evaluation with more recent network traffic with/without attacks. CSE-CIC-IDS-2018 dataset was generated by collecting ten days of network traffic, and system logs in ten subsets with different types of attacks using CICFlowMeter-V3 [49] and contains about 80 features for each record, including forward and backward directions of network flow and packets. Table 3 listed different types of attacks used in preparing the CIC-IDS-2018 dataset and the tools it used to generate the attack and attack duration. Qianru [50] analyzes the CSE-CIC-IDS-2018 dataset and pre-processes the dataset by eliminating normal and noisy ones and removing unnecessary values after the decimal point. After applying these techniques, the size of CSE-CIC-IDS-2018 decreased by 4MB, still 400GB way bigger than that of CSE-CIC-IDS-2017.

4.2 Data Pre-possessing

We pre-process each record into a string and fill up the NaN values with a default string. Afterwards, we generate tokens from the vocabulary and tokenize each input string by punctuation. To realize the large non-redundant data for training and verification, we employ a Stratified K-Fold Cross Validation strategy [51], which is also commonly used in machine learning. The scheme splits the dataset into k groups and from them, $k-1$ groups as a whole are used for training and the rest one group is used for validation and the strategy is also called *Leave One Out Strategy*.

One-hot-encoding technique is employed to convert labels into numerical forms. One-hot-encoding is a process in which categorical variables are converted into a form that could be provided to machine learning or deep learning algorithms to predict performance better.

5 Results and Performance Evaluation

In this section, we first compare the performance of our proposed model with different benchmark datasets. Then, we compare the performance of our model with the existing variations of our model. We mainly compare our work with the two most common variations which are CNN [24,26–28] and LSTM-CNN [9,12, 29–31,33] based models. We also compare our works with the works [12] that obtained the highest accuracy among existing works.

5.1 Evaluation Metrics

While we value the accuracy most, it has some serious flaws for imbalanced datasets like intrusion detection data. Accuracy is mainly used with a balanced dataset when a true positive and true negative is more important than identifying false positives (normal traffic predicted as anomalous traffic) and false negatives (anomalous traffic predicted as normal traffic). However, in an intrusion detection system, where the data is imbalanced, false positive and false negative count is also equally important to true positive and true negative count. In this scenario, calculating recall and F1-Score is more important than accuracy where F1-Score consider both the false positive and false negative count.

We evaluate our model in terms of the F1-Score and Recall score. We calculate standard machine learning algorithm performance metrics precision, recall, and F1-score with respect to True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) values as follows using Eqs. (1), (2) and (3):

Precision (ρ) [52]: It measures the proportion of the predicted attack that is actually an attack. Precision is calculated using the following equation.

$$\rho = \frac{TP}{TP + FP} \tag{1}$$

Recall (γ) [53]: It measures the proportion of actual attacks that are correctly identified and are computed as follows. We can also get the value of FNR simply by calculating recall.

$$\gamma = \frac{TP}{TP + FN}. = 1 - FNR \tag{2}$$

F1-Score (F1-Score) [54]: F1-Score is the harmonic mean of precision and recall. Validation accuracy is computed as the following equation.

$$F1 - Score = \frac{2 * \rho * \gamma}{\rho + \gamma} \tag{3}$$

5.2 Performance Analysis with Different Benchmark Datasets

We measure the performance against four different datasets, which are CSE-CIC-IDS-2018 [11], NSL-KDD [10], CSIC-2010 [32], and KDD-99 [21]. Table 4 summarizes our model's performance in terms of both F1-score and recall score.

The table also shows the comparison with other variants on the CSE-CIC-IDS-2018 dataset. In almost all cases, we operate on the whole dataset following 10-fold validation except 22-02-2018 and 23-02-2018, where the anomaly data is less than 0.0001% of the total data, and we have to work with a balanced dataset. It is evident from the table that we got 100% accuracy and recall scores from a few datasets, and for most of the datasets, the results are very promising.

In Table 5, the performance of our model on NSL-KDD data has been elaborated. We tested with different values of K-folds cross-validation for NSL-KDD data. We get a maximum F1-Score of 99.22% for 10-fold cross-validation, with a maximum recall of 99.97% for 8-fold cross-validation. Overall, 10-fold cross-validation provides a better result than others. Table 6 shows KDD-99 & CSIC-2010 data performance. Here our model clearly outperforms the LSTM-CNN based model [33] by a significant margin where they achieved the highest 94% accuracy by their proposed methods on the CSIC-2010 dataset. We obtain a low recall for the CSIC-2010 dataset because the dataset was primarily built for unsupervised learning while our approach is supervised. We also get a satisfactory result with the KDD-99 dataset and this outperforms all the existing works in terms of F1-Score (99.74%) and recall score (100%).

Table 4. Performance comparison of our proposed model using CSE-CIC-IDS-2018 dataset [11]

Dataset	Models					
	F1-Score			Recall		
	CNN	*CNN-LSTM*	*Proposed model*	*CNN*	*CNN-LSTM*	*Proposed model*
14-02-18	99.88	85.35	100.0	99.96	99.88	100.0
15-02-18	67.07	92.65	99.96	55.09	96.47	99.94
16-02-18	99.11	98.72	99.99	99.95	98.38	100.0
21-02-18	99.66	93.10	100.0	99.45	90.25	100.0
22-02-18	80.65	74.19	93.10	68.88	97.22	94.44
23-02-18	62.65	71.91	81.61	64.10	60.20	69.23
28-02-18	71.07	70.67	95.91	56.13	59.39	98.01
01-03-18	56.28	65.52	88.19	39.40	52.99	95.23
02-03-18	99.74	99.33	99.97	99.77	99.40	99.98

Table 5. Performance of the proposed model on NSL-KDD dataset [10]

Dataset	Metrics					
	F1-Score	*Precision*	*Recall*	*TNR*	*FPR*	*FNR*
2 fold	98.19	97.86	98.53	98.01	1.98	1.46
4 fold	96.00	92.37	99.94	96.51	3.49	0.05
6 fold	98.72	97.82	99.63	96.36	3.63	0.37
8 fold	99.12	98.30	99.97	97.1	2.89	0.22
10 fold	99.22	98.55	99.91	97.5	2.49	0.08

Table 6. Performance on CSIC-2010 [32] and KDD-99 [21] dataset

Dataset	Metrics		
	F1-Score	*Precision*	*Recall*
KDD-99	99.74	99.5	1.00
CSIC 2010	36.12	72.74	24.03

5.3 Performance Comparison with State-of-the-Art Works

In this section, we compare the performance of our proposed model with existing variations. We mainly compare our work with the two most common variations, which are CNN [24,26–28], and LSTM-CNN [9,12,29–31,33] based models. For comparing the performance of our model, we implement the state-of-the-art machine learning algorithms and use the most recent benchmark dataset CSE-CIC-IDS-2018 [11]. We also compare our works with the latest proposed model LuNet [12] which encompasses the highest proclaimed accuracy using the NSL-KDD dataset [10].

Figure 2 shows that our proposed model outperforms the nearest variants in terms of F1-Score using the CSE-CIC-IDS-2018 dataset [11]. At the same time, our model achieves a higher recall score which is nearly optimum, as shown in Fig. 3. From the results, it is also evident that normal CNN performs better than the CNN-LSTM models. We get comparatively poor results on the dataset of 22-02-2018 and 23-2018 because there were less anomalous or attacked data on the dataset, and we had to use a balanced dataset. But the overall performance of our model is superior to the nearest variations, and our model provides nearly optimal results. Thus, it easily proves the efficiency of our model for detecting known attacks with a higher F1-Score and Recall value.

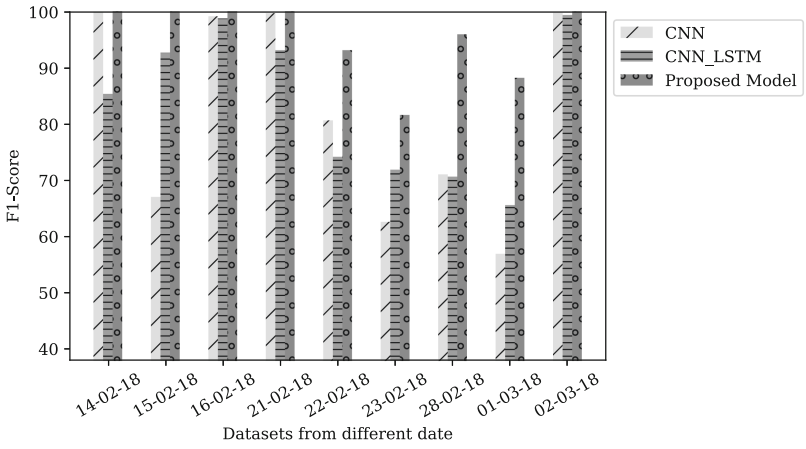

Fig. 2. Comparing F1-Score between our model and State-of-the-art CNN [24,26–28], and CNN-LSTM [9,12,29–31,33] models using CSE-CIC-IDS-2018 dataset [11]

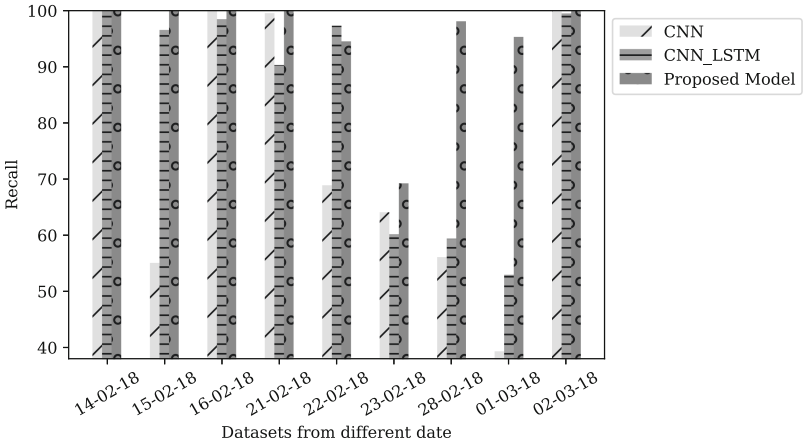

Fig. 3. Comparing recall between our proposed model and State-of-the-art CNN [24, 26–28], and CNN-LSTM [9,12,29–31,33] model using CSE-CIC-IDS-2018 dataset [11]

We compare the performance of our model with LuNet [12] using the NSL-KDD dataset [10] as LuNet does not use the latest CSE-CIC-IDS-2018 dataset [11]. From Fig. 4, it can be seen that our model closely follows LuNet by the F1-Score, achieving almost similar or even better in the 10-fold dataset. It is also worth mentioning that our proposed model is a lightweight model with a few layers compared to LuNet.

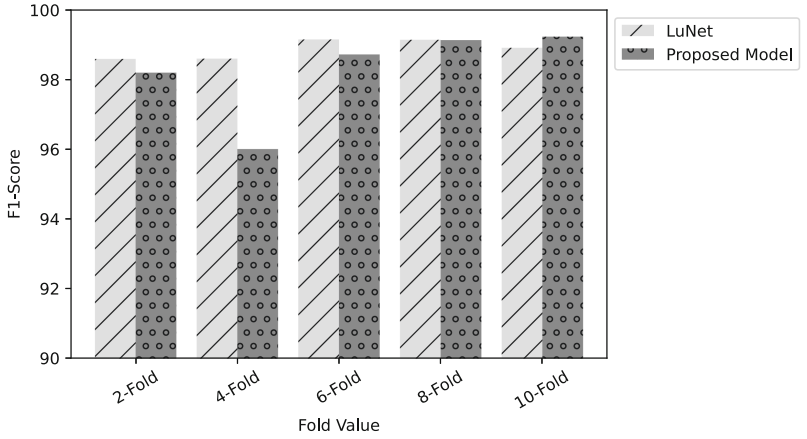

Fig. 4. Performance comparison based on F1-Score between LuNet [12] and our model

Figure 5 shows the comparison of Recall score between our proposed model and LuNet [12]. It is very prominent from the figure that our proposed model

outperforms LuNet [12] except for 2-fold cross-validation data. So, we can conclude that our proposed model provides a better recall score than the existing models, which ensures the efficiency of IDS.

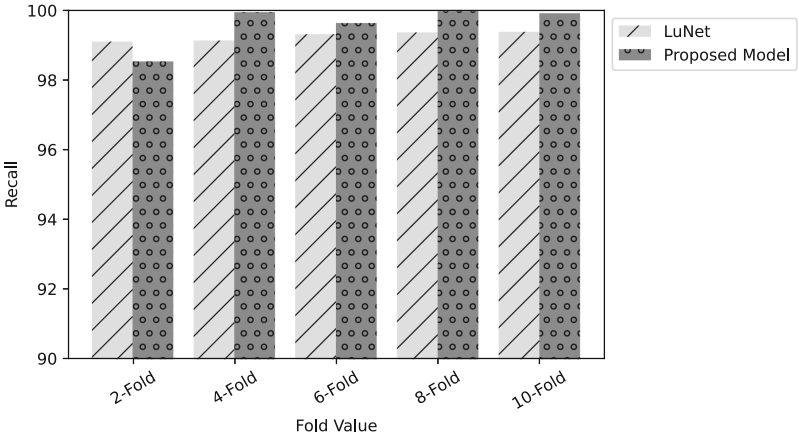

Fig. 5. Performance comparison based on Recall between LuNet [12] and our model

6 Conclusion

In this paper, we investigated the potential of combining the CNN, LSTM, and GRU models with attention mechanisms for improving the accuracy of intrusion detection systems. We evaluated the performance of our proposed model with the most recent and state-of-the-art benchmark network traffic datasets, such as NSL-KDD [10] and CSE-CIC-IDS-2018 [11] comprising of various attacks. The evaluation shows the efficiency of our proposed model with a higher F1-score and Recall score for all datasets. For the CSE-CIC-IDS-2018 dataset, we get an almost optimal result. We also showed the comparison of our proposed model with the existing neural network based models (e.g., an average F1-Score value of 95.41% by our model vs. 81.87% for CNN and 83.50% for CNN-LSTM). Even though we built a model with optimal F1-Score and Recall value, we have considered anomaly detection as a binary classification problem where all the anomalous data or attacks are considered in only one class. In the future, this can be generalized for multi-class classification and detect different anomalies or attacks individually. Our works can also be extended with different hyper-parameter values of CNN (i.e., activation function, loss function, learning rate, etc.). The impact of adding more layers to CNN and LSTM or using bi-directional LSTM could also be investigated in the future. Currently, the detection is offline, it could be make online as an application to deal with online data.

Acknowledgment. The authors thank the anonymous reviewers for their comments. This material is based upon work partially supported by the Natural Sciences and Engineering Research Council of Canada under Discovery Grant N02815.

References

1. CISCO: Global 2021 forecast highlights (2021). https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf. Accessed 30 June 2021
2. IoT Business News: Global IoT roaming data traffic to increase by 300% to reach 500pb in 2025 (2021). https://iotbusinessnews.com/2020/10/15/70310-global-iot-roaming-data-traffic-to-increase-by-300-to-reach-500pb-in-2025/. Accessed 30 June 2021
3. CISCO: Cisco annual internet report (2018–2023) white paper (2021). https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Accessed 30 June 2021
4. Biermann, E., Cloete, E., Venter, L.M.: A comparison of intrusion detection systems. Comput. Secur. **20**(8), 676–683 (2001)
5. Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., Lin, W.-Y.: Intrusion detection by machine learning: a review. Expert Syst. Appl. **36**(10), 11994–12000 (2009)
6. Zhang, Z., Li, J., Manikopoulos, C.N., Jorgenson, J., Ucles, J.: HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In: Proceedings of the IEEE Workshop on Information Assurance and Security, vol. 85, p. 90 (2001)
7. Vinayakumar, R., Soman, K.P., Poornachandran, P.: Applying convolutional neural network for network intrusion detection. In: ICACCI. IEEE (2017)
8. Botros, S.M., Diep, T.A., Izenson, M.D.: Method and apparatus for training a neural network model for use in computer network intrusion detection. US Patent 6,769,066, 27 July 2004
9. Hsu, C.-M., Azhari, M.Z., Hsieh, H.-Y., Prakosa, S.W., Leu, J.-S.: Robust network intrusion detection scheme using long-short term memory based convolutional neural networks. Mob. Netw. Appl. **26**(3), 1137–1144 (2020). https://doi.org/10.1007/s11036-020-01623-2
10. Canadian Institute for Cybersecurity (CIC): NSL-KDD dataset (2009). https://www.unb.ca/cic/datasets/nsl.html. Accessed 30 June 2021
11. Canadian Institute for Cybersecurity (CIC): CSE-CIC-IDS2018 on AWS (2018). https://www.unb.ca/cic/datasets/ids-2018.html. Accessed 30 June 2021
12. Wu, P., Guo, H.: LuNET: a deep neural network for network intrusion detection. In: SSCI. IEEE (2019)
13. Tjhai, G.C., Papadaki, M., Furnell, S.M., Clarke, N.L.: Investigating the problem of IDS false alarms: an experimental study using snort. In: Jajodia, S., Samarati, P., Cimato, S. (eds.) SEC 2008. ITIFIP, vol. 278, pp. 253–267. Springer, Boston, MA (2008). https://doi.org/10.1007/978-0-387-09699-5_17
14. KirstenS, Wichers, Jkurucar, kingthorin: Intrusion detection control-OWASP (2021). https://owasp.org/www-community/controls/Intrusion_Detection. Accessed 30 June 2021
15. Liu, C., Liu, Y., Yan, Y., Wang, J.: An intrusion detection model with hierarchical attention mechanism. IEEE Access **8**, 67542–67554 (2020)
16. Yan, L., Xiong, J.: Web-APT-Detect: a framework for web-based advanced persistent threat detection using self-translation machine with attention. IEEE Lett. Comput. Soc. **3**(2), 66–69 (2020)
17. Liu, T., Qi, Y., Shi, L., Yan, J.: Locate-then-detect: real-time web attack detection via attention-based deep neural networks. In: IJCAI, pp. 4725–4731 (2019)

18. Shun, J., Malki, H.A.: Network intrusion detection system using neural networks. In: ICNC, vol. 5, pp. 242–246. IEEE (2008)
19. MIT: 1999 DARPA intrusion detection evaluation dataset (1999). https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset. Accessed 30 June 2021
20. Chiba, Z., Abghour, N., Moussaid, K., El Omri, A., Rida, M.: A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection. Comput. Secur. **75**, 36–58 (2018)
21. KDD 1999: KDD cup 1999 data (2021). http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed 30 June 2021
22. Mahalingam, P.R.: Intelligent network-based intrusion detection system (iNIDS). In: Meghanathan, N., Nagamalai, D., Chaki, N. (eds.) Advances in Computing and Information Technology. Advances in Intelligent Systems and Computing, vol. 176, pp. 1–9. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31513-8_1
23. Su, L., Yu, L., Li, T., Liu, X.: Research on network data security based on RS-PS Support Vector Machine (SVM). J. Phys: Conf. Ser. **1748**(3), 032057 (2020). IOP Publishing
24. Mendonça, R.V., et al.: Intrusion detection system based on fast hierarchical deep convolutional neural network. IEEE Access **9**, 61024–61034 (2021)
25. Canadian Institute for Cybersecurity (CIC): Intrusion detection evaluation dataset (CIC-IDS2017) (2017). https://www.unb.ca/cic/datasets/ids-2017.html. Accessed 30 June 2021
26. Kim, J., Kim, J., Kim, H., Shim, M., Choi, E.: CNN-based network intrusion detection against denial-of-service attacks. Electronics **9**(6), 916 (2020)
27. Wang, H., Cao, Z., Hong, B.: A network intrusion detection system based on convolutional neural network. J. Intell. Fuzzy Syst. **38**(6), 7623–7637 (2020)
28. Bandyopadhyay, S., Chowdhury, R., Roy, A., Saha, B.: A step forward to revolutionise intrusiondetection system using deep convolution neural network. Preprints (2020)
29. Sun, P., et al.: DL-IDS: extracting features using CNN-LSTM hybrid network for intrusion detection system. Secur. Commun. Netw. **2020**, Article ID: 8890306, 11 (2020). https://doi.org/10.1155/2020/8890306
30. Kim, J., Kim, J., Thu, H.L.T., Kim, H.: Long short term memory recurrent neural network classifier for intrusion detection. In: PlatCon, pp. 1–5. IEEE (2016)
31. Kuang, X., et al.: DeepWAF: detecting web attacks based on CNN and LSTM models. In: Vaidya, J., Zhang, X., Li, J. (eds.) CSS 2019. LNCS, vol. 11983, pp. 121–136. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-37352-8_11
32. CSIC: HTTP dataset CSIC 2010 (2010). https://www.tic.itefi.csic.es/dataset/. Accessed 30 June 2021
33. Hsu, C.-M., Hsieh, H.-Y., Prakosa, S.W., Azhari, M.Z., Leu, J.-S.: Using long-short-term memory based convolutional neural networks for network intrusion detection. In: Chen, J.-L., Pang, A.-C., Deng, D.-J., Lin, C.-C. (eds.) WICON 2018. LNICST, vol. 264, pp. 86–94. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-06158-6_9
34. UNSW: The UNSW-NB15 dataset (2015). https://research.unsw.edu.au/projects/unsw-nb15-dataset. Accessed 30 June 2021
35. Basati, A., Faghih, M.M.: APAE: an IoT intrusion detection system using asymmetric parallel auto-encoder. Neural Comput. Appl. 1–21 (2021). https://doi.org/10.1007/s00521-021-06011-9

36. Sekhar, R., Sasirekha, K., Raja, P.S., Thangavel, K.: A novel GPU based intrusion detection system using deep autoencoder with Fruitfly optimization. SN Appl. Sci. **3**(6), 1–16 (2021)
37. Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Comput. Intell. **2**(1), 41–50 (2018)
38. Tang, R., et al.: ZeroWall: detecting zero-day web attacks through encoder-decoder recurrent neural networks. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 2479–2488. IEEE (2020)
39. Vartouni, A.M., Kashi, S.S., Teshnehlab, M.: An anomaly detection method to detect web attacks using stacked auto-encoder. In: 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), pp. 131–134. IEEE (2018)
40. Mac, H., Truong, D., Nguyen, L., Nguyen, H., Tran, H.A., Tran, D.: Detecting attacks on web applications using autoencoder. In: Proceedings of the Ninth International Symposium on Information and Communication Technology, pp. 416–421 (2018)
41. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
42. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015)
43. Keras: Keras (2020). https://keras.io/. Accessed 30 June 2021
44. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
45. Keras: Nadam (2015). https://keras.io/api/optimizers/Nadam/. Accessed 30 June 2021
46. Lippmann, R.P., et al.: Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In: Proceedings of the DARPA Information Survivability Conference and Exposition, DISCEX 2000, vol. 2, pp. 12–26. IEEE (2000)
47. McHugh, J.: Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln laboratory. ACM Trans. Inf. Syst. Secur. (TISSEC) **3**(4), 262–294 (2000)
48. Kaushik, S.S., Deshmukh, P.R.: Detection of attacks in an intrusion detection system. Int. J. Comput. Sci. Inf. Technol. (IJCSIT) **2**(3), 982–986 (2011)
49. The Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC): Cicflowmeter (formerly iscxflowmeter) (2021). https://www.unb.ca/cic/research/applications.html. Accessed 30 June 2021
50. Zhou, Q., Pezaros, D.: Evaluation of machine learning classifiers for zero-day intrusion detection-an analysis on CIC-AWS-2018 dataset. arXiv preprint arXiv:1905.03685 (2019)
51. scikit-learn.org: sklearn.model_selection.stratifiedkfold (2020). https://scikit-learn.org/stable/. Accessed 30 June 2021
52. scikit-learn.org: sklearn.metrics.precision_score (2021). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html. Accessed 30 June 2021
53. scikit-learn.org: sklearn.metrics.recall_score (2021). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html. Accessed 30 June 2021
54. scikit-learn.org: sklearn.metrics.f1_score (2021). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html. Accessed 30 June 2021