electronics

MDPI

Article

Performance Evaluation of Deep Learning Based Network Intrusion Detection System across Multiple Balanced and Imbalanced Datasets

Azizjon Meliboev [1], Jumabek Alikhanov [2] and Wooseong Kim [1,*]

[1] Computer Engineering Department, Gachon University, Seongnam 13120, Korea; meliboyevabdulaziz@gmail.com
[2] Computer Science and Information Engineering Department, Inha University, Incheon 22212, Korea; jumabek@nsl.inha.ac.kr
* Correspondence: wooseong@gachon.ac.kr

Abstract: In the modern era of active network throughput and communication, the study of Intrusion Detection Systems (IDS) is a crucial role to ensure safe network resources and information from outside invasion. Recently, IDS has become a needful tool for improving flexibility and efficiency for unexpected and unpredictable invasions of the network. Deep learning (DL) is an essential and well-known tool to solve complex system problems and can learn rich features of enormous data. In this work, we aimed at a DL method for applying the effective and adaptive IDS by applying the architectures such as Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU). CNN models have already proved an incredible performance in computer vision tasks. Moreover, the CNN can be applied to time-sequence data. We implement the DL models such as CNN, LSTM, RNN, GRU by using sequential data in a prearranged time range as a malicious traffic record for developing the IDS. The benign and attack records of network activities are classified, and a label is given for the supervised-learning method. We applied our approaches to three different benchmark data sets which are UNSW NB15, KDDCup '99, NSL-KDD to show the efficiency of DL approaches. For contrast in performance, we applied CNN and LSTM combination models with varied parameters and architectures. In each implementation, we trained the models until 100 epochs accompanied by a learning rate of 0.0001 for both balanced and imbalanced train data scenarios. The single CNN and combination of LSTM models have overcome compared to others. This is essentially because the CNN model can learn high-level features that characterize the abstract patterns from network traffic records data.

Keywords: CNN; RNN; LSTM; GRU; IDS; UNSW NB15; KDDCup '99; NSL-KDD

Citation: Meliboev, A.; Alikhanov, J.; Kim, W. Performance Evaluation of Deep Learning Based Network Intrusion Detection System across Multiple Balanced and Imbalanced Datasets. *Electronics* **2022**, *11*, 515. https://doi.org/10.3390/electronics11040515

Academic Editor: Nour Moustafa

Received: 19 January 2022
Accepted: 6 February 2022
Published: 9 February 2022

1. Introduction

Due to the high demand for internet connectivity between computers, IDS becomes a vital application for network security to inspect various invasion behavior in networks. In our work, we developed network-based IDS which inspects all coming packets and determines any distrustful behavior. In the deployment scenario once malicious traffic is identified, IDS notifies admins/firewalls or intrusion prevention systems. According to methodologies, network-based IDS is divided into two methods which are Signature-detection and Anomaly-detection. Signature detection works with pre-defined signatures and filters. This technique can inspect the defined intrusions effectively while an undefined attack record is not well determined. In contrast, the anomaly-detection technique relies on heuristic methods to detect the undefined attack behavior. That means when the system identifies a difference from benign traffic patterns then this traffic count as network intrusions. Recent studies showed that a heuristic method is a reliable approach for IDS. The heuristic approach is a proactive and strong method that can use ML and DL

algorithms such as unsupervised and supervised. The DL-based system has the ability to extract complex features from a huge amount of network traffic data and can learn the characterization of that complex features. Therefore, the DL based approach can solve the network security tasks and develop a reliable and flexible IDS. The amount of training data and features of data are essential for DL as well. In our work, we used traffic data that was gathered and labeled as either normal or abnormal traffic records from various network resources. DL algorithms can learn reliable feature representations of the data, which can deal with a supervised classification to achieve noticeable outcomes. In our previous work [1] we applied CNN with various architecture and combinations of LSTM and we achieved considerable results. Inspired by our previous work and the promise of DL, we improved and expanded our work by applying various DL models for various datasets. In fact, we used CNN [2,3], LSTM [4], RNN [5,6] and GRU [7] for supervised-learning task with batch normalization and multinomial logistic regression technique. Our deep learning model architectures are depicted in Figures 1–3. We aimed to show the comparison of various DL models evaluation by three different datasets which are UNSW NB15 [8,9], KDDCup '99 [10,11], NSL-KDD [12]. The remainder of the paper is arranged as follows. The Related Works section introduces related works which applied DL algorithms for intrusion detection problems. The Proposed Methods section details our proposed method of DL and its network architecture. The section of Data Description detailed all datasets and presented data imbalance. The section of Experimental Results described details of our models, solving an imbalanced problem, and all model results. The last section concludes the paper.

2. Related Work

The traditional ML algorithms used to develop IDSs, such as K-Nearest Neighbors (KNN) [13], Naive-Bayesian (NB) [14], Random Forests (RF) [15]. The ML algorithms are applied to distinguish the normal behavior from the abnormal behavior as classifiers. The many algorithms for IDS accomplish a feature selection technique to extract patterns of proper characters from the network traffic records data to increase the performance of classification. Recently, applications of DL algorithms have been experimented with audio, speech, and image processing with success and showed unbelievable performance. The DL methodologies intend to learn good feature representations from a huge number of unlabeled data for an unsupervised-learning method to solve such as clustering and association problems, and a huge amount of labeled data is for a supervised-learning method to solve classification and regression tasks. The supervised learning method ensures that machines can train a labeled (defined) group of input data, and on that basis, machines make predictions.

In network security, DL has applied successfully for IDS and provided the expected solution with remarkable performance. According to the results of recent studies, DL-based IDS prevailed over traditional approaches. Yadav Balram et al. [16] represented that comparison of recent research for applying IDS to conventional IDS methods and DL approaches.

Javaid et al. [17] claim a DL method to build an efficient and adaptable IDS in their work. They used the self-taught learning (STL) technique which associates sparse auto-encoder with multinomial logistic regression. The authors evaluated their model in the NSL-KDD dataset. STL reached an accuracy score of 88.38% for the binary classification and 79.10% for the multi-class classification.

Shone et al. [18] represented their supervised feature learning approach by applying a DL-based Non-Symmetric Deep Auto-encoder (NDAE) algorithm. This approach has been evaluated in NSL-KDD and KDD Cup '99 datasets. They reveal that the approach reaches an accuracy score of 85.42% in multi-class (5) and 89.22% in multi-class (13) classification in the NSL-KDD dataset.

Yin et al. [19] aimed using the DL algorithm which is RNN for IDS. For the network, input is as 122-dimensional input neurons and output 2 as binary classification. The authors

declare that their model reached an accuracy score of 83.28% in binary-classification (2) and 81.29% multi-classification (5) when evaluated in NSL-KDD dataset.

Vinayakumar et al. [20] reveal that they used several DL algorithms for IDS in their work. They used combination models which are CNN-LSTM, CNN-RNN, CNN-GRU. The combination model which is CNN-LSTM reaches an accuracy score of 99.7% in binary (2) classification and 98.7% in multi-class (5) classification when evaluated in KDDCup '99 dataset. Other performances are also remarkable rather than other existing works with this dataset.

Wang et al. [21], purposed intrusion detection classification by using DL-based representation learning method. This method is the recently developed ML approach that directly analysis raw network activity data; This approach automatically learns features from raw data. The accuracy of the model is not very good because of the low image size. The authors also implemented a 5×5 filter size with 16 and 32 filters in their CNN architecture.

Hsiao et al. [22], reveal their model that is Siamese CNN (SCNN). DL models require a large number of data (many training samples) for remarkable performance. For visualizing the malware image authors used the image processing technique. The proposed model was successfully applied due to the big image size of malware and the architecture of Siamese CNN has a sequence of twin CNN.

Cui Zhihua et al. [23] proposed to develop the detection of intrusions by using DL methods and also solved imbalanced data problems. The authors first focused on visualizing the malicious code into images to then classify using a CNN. The implementation run with different sizes of malware images (24*24, 48*48, 96*96, 192*192) and also using different architectures of CNN respectively. The main assets of their method are that they solved imbalanced data problems and showed a good detection speed. Due to the architecture of very deep CNN and also resolved the imbalanced data problem, the model successfully performed with an accuracy score of 94.5%.

Although notable progress has been made to improve the predictive ability of NIDSs in the above-mentioned research, a recent study that investigated the impact of packet sampling techniques on NIDS models brought new insight [24]. The study demonstrated that even the smallest sampling rates such as 1/100 and 1/1000 drastically reduce the performance of ML-based NIDS systems. The importance of addressing training data imbalance is also highlighted in [25]. Defensive approaches against adversarial attacks on has also been studied recently [26,27]. Deep learning is also being actively researched on IoT, wireless sensor networks and edge devices [28–30].

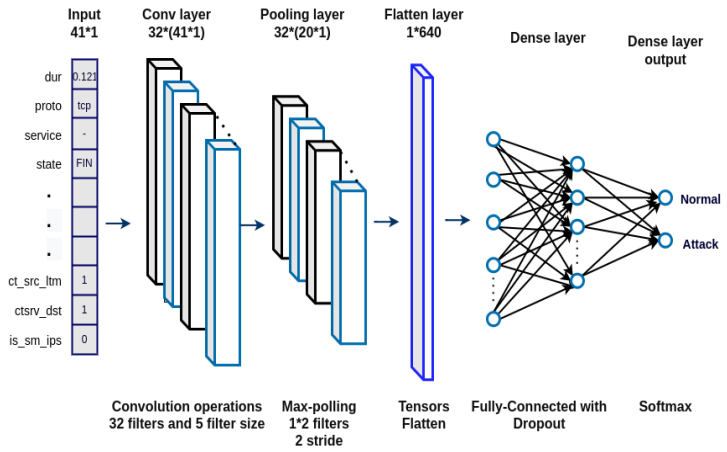

Figure 1. The single CNN architecture. The network includes following layers: input layer, one convolution layer, one max-pooling layer, two fully connected layers and output layer.

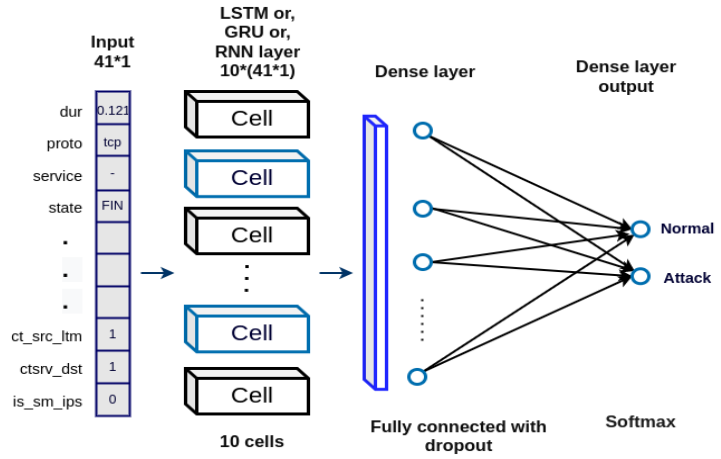

Figure 2. The three architectures which are LSTM, GRU, RNN models are illustrated. Except LSTM, GRU, RNN layers, other layers same as following layers: input layer, LSTM, or GRU, or RNN layers, Dense layer, fully connected layer and output layer.

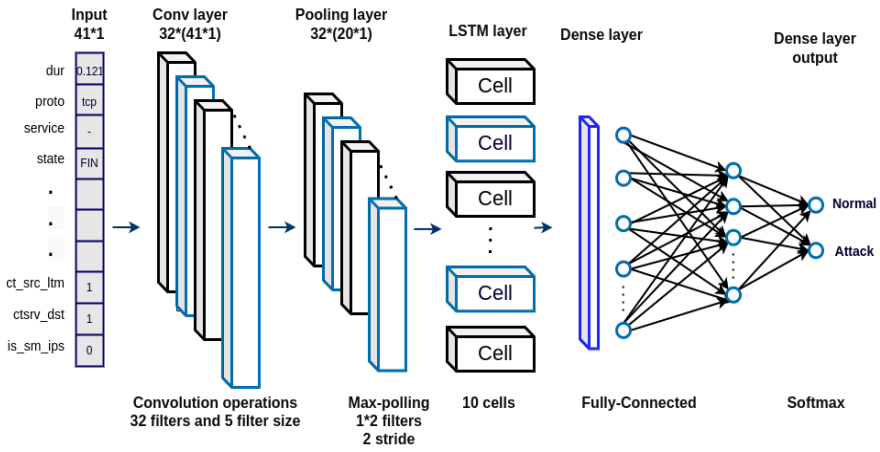

Figure 3. The architecture of the CNN and LSTM combination model. The architecture includes following layers: input layer, one convolution layer, one max-pooling layer, one LSTM layer, Dense layer, fully connected layer and output layer.

3. Proposed Method

DL algorithm CNN was successfully solved computer vision tasks such as image classification, detection of objects, face identification, pattern recognition. The CNN is proposed to reduce the images into a shape that is easier to manipulate, without losing features which are essential for reaching a remarkable performance of the model. Convolution operations consist of filters (kernels) that can extract complex features of patterns from the image. In our previous work, we have applied this CNN algorithm to dissimilar data, which was one-dimensional numeric array data. We have supposed the feature map matrix by the time-sequence of feature vectors as an image. The feature vectors associate with time-sequence and represent corresponding local features, which led us to appropriate this relationship. Inspiring our previous research, we aimed to apply our models to other datasets and implemented other DL algorithms as well.

According to DL models, we approach various architectures which are represented in Figures 1–3. Figure 1 illustrated the architecture of well-trained CNN model. In the convolution layer, we used input as 41*1, 32 convolution kernels, the size of the kernel is 5, and a 'Sigmoid' activation function. One kernel is able to learn one particular feature in the first convolution layer. That is inadequate, for that reason we set up 32 kernels that led us to generate 32 distinctive features on the first convolution layer. After the first convolution layer, a 37×32 feature map matrix is generated. Every kernel consists of its weight with

set out kernel size, the reflection of the length of the feature map. Max-pooling [31] is a layer that is a type of convolution layer that is accepted and allows down sample output. A filter chooses the highest value of the region to which the filter is covered in the max-pooling layer. It cut down the dimension size of the output and decreases the number of features and complexity of the network. We applied the max-pooling technique with 2 pooling sizes and 2 stride sizes. In the flattened layer, a tensor is reshaped which is equal to the size of elements containing 1×640 dimensions. In general, fully connected layers contain learnable parameters and neurons connected to each other. Therefore, we add Dense layers are for two fully-connected layers to execute the classification with a dropout technique [32] which we set 0.4 and a batch normalization [33] which we set 32. At the end of the network, we used the softmax activation [34] function to prevent over-fitting which calculates the probability for each class. In the training process, a dropout provides regularization where neurons are dropped out randomly when output comes from the previous layer. For preventing over-fitting and improving generalization this technique plays a role and ensures the network does not over-rely on any specific input. To make training convergence faster and more stable the batch normalization technique is used. During the training, each feature in the batch is re-scaled again later to complete the whole training dataset. The learned again means that variation replaces the ones achieved at batch-level.

Figure 2 illustrated three types of architectures of the model. We only changed LSTM, GRU and RNN and left other layers or hyperparameters. These recurrent algorithms have units that mean dimensions of inner cells where we set 100 smart neurons in each layer. Then we used one Dense layer which is for normalizing batch and one dropout technique as well. At the end of the network, one dense layer acts as a fully-connected layer with a softmax activation function.

The combination model is represented in Figure 3. According to the experiment, LSTM corresponded well with the combination model. After one convolution operation with 32 filters, 5 filter size, and one pooling with 2 filters, 2 stride size operation, we used the LSTM layer with 100 cells which performed well. Then batch normalization layer and dropout technique as well. Same as previous architecture, at the end of the network we set a fully-connected layer with softmax activation function for probability of output.

4. Dataset Description

We used three different datasets for our model evaluation. In previous work we evaluated our model in UNSW_NB15 dataset but in this work, we aimed to apply KDD_cup99 and NSL-KDD datasets. The UNSW_NB15 intrusion dataset is generated in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for creating a mixture of real normal behaviors and artificial attack behaviors. This dataset was evaluated widely in experiments of IDS. The attack classes of the dataset have 9 types, namely, Generic, Reconnaissance, Fuzzers, Backdoors, DoS, Analysis, Exploits, Shellcode, and Worms. The features of the dataset generated 45 features with a label. This dataset is set up labeled as attack and normal categories respectively 0 if the activity is normal and 1 if the activity is attacked. For comparative study and to show the reliability of DL models, we evaluated our proposed models KDD_cup99 and NSL-KDD datasets. This is the main reason why we chose datasets that have been used in the evaluation of network flow analysis of intrusion detection to the highest extent. Stolfo, et al. The KDD Cup 1999 data was established by the data captured in the DARPA IDS evaluation procedure. This dataset is older but can ensure a good comparison for each IDS model. There are 4,898,431 network records in the dataset and each record has 41 features. According to record features, there are 22 attack categories but not proportional where one category attack is highest among all attack records or other one attacks only accounted for 1%. Therefore, each record of training and test sets is imbalanced. For the reason of imbalance, we applied binary classification that detects attacks and normal records.

The dataset of NSL-KDD has a reduced amount of data and improved generation of the KDD_cup99 dataset for the evaluation of models of IDS. It is considered that the test data is not from the same probability allocation as the training data, and it includes specific attack types not in the training data. The analysis of KDD 99 showed that about 78% and 75% of the traffic activities are duplicated in the train and test set, respectively. This highly affects the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches. To solve these issues, an improved dataset known as NSL-KDD was created by removing all redundant (duplicate) instances. This reduces the size of the dataset, however, does not reduce the performance of the algorithm as only duplicate unnecessary records are removed. In other words 78% and 75% duplicate records which were not useful are excluded from KDD 99. According to categorically classification, the NSL-KDD dataset heavily imbalanced dataset where the dataset contains 53% of normal class and attack classes contributed 0.78% of Remote-to-Local (R2L) class, 0.041% of User-to-Root (U2R) class, and other minor attack classes but when we apply binary classification it is not imbalanced. These imbalances of the dataset highly affect the performance of the classifier in the detection of the minor classes. Therefore, the binary classification method is suitable to evaluate our models.

A partition from this dataset is configured as a training set and testing set. Due to the amount of attack and normal being quite imbalanced, we used a Synthetic Minority Oversampling Technique (SMOTE) [35]. We give particulars of this technology in Section 5. The normal and attack activities as shown in Table 1.

Table 1. Distribution after Data split.

Dataset	Data Split	Number of Normal	Number of Attack	Total
UNSW_NB15	Train	56,000	119,341	175,341
	Balanced Train	119,341	119,341	238,682
	Test	37,000	45,332	82,332
KDD_cup99	Train	97,277	396,743	496,020
	Balanced Train	396,743	396,743	793,486
	Test	60,592	250,436	311,028
NSL-KDD	Train	67,343	58,630	125,973
	Balanced Train	67,343	67,343	134,686
	Test	9711	12,833	22,544

Data Preprocessing

According to the characteristic of the UNSW_NB15 dataset, the numeric features that integers are 41 and non-numeric features that strings are 3. Properly, we only can train our DL models with the numeric input value and therefore we have to transform non-numeric features into numeric features. Similarly, in case of other KDD_cup99 and NSL-KDD, non-numeric features are processed. Moreover, we set the data type of array as float type and removed the features such as 'id', 'string label'.

5. Experimental Results

All experiments are executed by using the Keras [36], TensorFlow [37] open-source software library and Numpy, Scikits Learn (Sklearn), Panda machine learning libraries that provide the Python3 interface on the Ubuntu 20.04 operating system. We trained all models on 8 GB local GPU memory. For comparative studies, we also implemented many layer models of CNN using Keras.

5.1. Experiment of Models

CNN has the ability to identify proper features from our data and can learn that proper features. The network included an input layer, hidden layer, and output layer with respective criteria. Our implementations went on two tracks that used balanced and

imbalanced data. Although we have executed the 32, 64, and 128 filters with filter sizes 3, 5, and 10, the results are not noticeable. Our best parameters for the convolution operation are 32 filters, the size of filter 5, and the size of batch 32 with the "Sigmoid" activation function. In the pooling layer, we set up a max-pooling technique with the size of pooling 2 and the size of stride 2. Also, we used Batch normalization where 32 batch size with "sigmoid" activation function and Dropout 0.5. According to the performance of the model with various learning rates of 0.01, 0.001, and 0.0001, we used the Adam optimizer with a learning rate of 0.0001 and run CNN model experiments until 100 epochs. However, the recurrent models LSTM, GRU, and RNN are quite similar. The models run 0.001 learning rate due to sequence computation with many recurrent blocks and heavy training time consumption. To see the optimal performance, we increased the number of recurrent blocks, which is 10 to 100, but the results are not too much different. In contrast, training time consumption is increased. In the case of the combination model, the parameters of CNN are the same as the single CNN model that has 32 filters and the size of the filter 5. We only combined with the LSTM model which has 10 recurrent blocks due to poor performance with other recurrent models. We decided to set 32 filters, the size of filter 5 for the convolution operation for the CNN model and 10 recurrent blocks for the LSTM, GRU, RNN models respectively Adam optimizer with 0.0001 and 0.001 learning rate.

We evaluated our IDS models on three different datasets which are UNSW_NB15, KDD_cup99, and NSL-KDD. According to imbalanced classes, we decided to apply binary classification that identifies attacks (all classes of attacks counted as attack 1) among normal (0) records. To show how imbalanced data affect performance, we trained two tracks of implementations that use balanced and imbalanced data with the above network structures until 100 epochs. According to all implementations, we can achieve the following considerations on those DL models.

- CNN model performs noticeably in terms of accuracy, precision, recall, and f-score on all three datasets.
- CNN model has trained with height results rather than other models.
- LSTM model showed remarkable performance as well
- CNN and LSTM combination model has required time and more epochs to reach adequate results.
- With GRU and RNN combination model does not show improvement in performance with imbalanced data of UNSW_NB15, but with balanced data, it works well
- More data and balanced data effect model performance

5.2. Parameters of Models

We experimented with various layers of convolution for CNN but an acceptable trained model included 2 convolution layers, 2 max-pooling layers, a dropout layer, batch normalization, and two fully-connected layers. According to features of UNSW_NB15 data, a shape of input is 42*1 and comes in a convolution layer. The CNN creates a tensor of shape 42*1*32 (filters 32) and it came in a max-pooling layer (size of filter 2 and size of stride 2). The Max-Pooling technique cut down the tensor shape into 20*1*32. Decreased tensor comes in convolution operation again and after that executed pooling operation again as above. Then all tensors flatten that removing all of the dimensions except for one. In the fully-connected layer all neurons are connected to each other. We used batch normalization (batch size is 32) for training faster and stable and a dropout technique (0.5) that ensures regularization by dropping inactive neurons where the previous layer output to zero. At the end of the network, we used the "Softmax" function for probability. Due to the features of KDD_cup99 and NSL-KDD being 41, input for the convolution layer was as 41*1*32 (number of filters 32). All other structures were the same as we explained above.

The recurrent models RNN, LSTM, GRU are very similar architectures. When we train our recurrent models, we increased recurrent cells (units) to 100, but it does not affect our experiment too much. After recurrent layer, we used batch normalization layer and dropout technique then "softmax" function for probability.

For the combination model, we decided to combine CNN and LSTM models. As CNN structure, we set one convolution and one pooling layer then we used LSTM cells. The batch normalization, dropout, and "softmax" used the same as other models.

5.3. Results

In the first track implementations, all models were trained and evaluated on the imbalanced data as represented in Table 2. According to the results, the model of CNN performed best when we evaluated all the datasets such as UNSW_NB15, KDD_cup99, and NSL-KDD, with accuracy scores of 85.8, 92.3, and 78.8, respectively. Moreover, the LSTM and hybrid model achieved noticeable results in all datasets, but they require more time to train. The models GRU and RNN did not show comparable performance when we feed UNSW_NB15 data.

Table 2. First Experiment Results.

	Accuracy	**Precision**	**Recall**	**F-Score**
KDD_cup99				
CNN	92.3	99.8	91	95.2
LSTM	91.8	98.6	91.1	94.7
GRU	90.7	99.7	88.7	93.8
RNN	91.7	99.4	90.2	94.6
CNN + LSTM	92.7	99.8	91	95.2
NSL-KDD				
CNN	78.8	96.7	65	77.7
LSTM	76.2	96.9	60.2	74.2
GRU	72.5	98.7	52.4	68.5
RNN	73.2	94.6	56.2	70.5
CNN + LSTM	85.5	96.1	77.1	85.9
UNSW_NB15				
CNN	85.8	80.9	99.4	87.8
LSTM	84.9	79.2	98.3	87.7
GRU	57	56.3	97.3	71.3
RNN	55	55.1	100	71
CNN + LSTM	80.8	74.4	99.3	85

In second track implementations in Table 3, Due to imbalanced classes, we used SMOTE technique that oversampled the instance in the minority class. That means simply duplicating instances from the minority class in the training dataset. As we mentioned above, we considered all attack classes as one class (1) and a normal class (0) and balanced these two classes that make an equal amount of attack and normal. According to the outcomes of the experiment, models performed much better than previous track experiments as illustrated in the Table 3. In this track, the CNN trained best on UNSW_NB15 and KDD_cup99 dataset respectively accuracy score of 91.2 and 95.2 but on NSL-KDD dataset combination model overcame with an accuracy score of 82.6. Other models also showed much better improvement particularly GRU and RNN significantly raised when model trains on UNSW_NB15 data rather than previous trail experiment respectively accuracy score of 77.9, 71.9.

The best train and test accuracy scores of the CNN model on the KDD_cup99 dataset are illustrated in Figure 4. Here the curve of training accuracy rate reaches roughly 100% after about 20 epochs and the curve of test accuracy around 95.3%. The GRU, RNN, and combination models are required more training time consumption rather than CNN because of models recurrently and sequentially computing and also the complexity of network.

Table 3. Second Experiment Results.

	Accuracy	Precision	Recall	F-Score
KDD_cup99				
CNN	95.2	99.5	90.7	94.9
LSTM	95.4	99.4	91.4	95.1
GRU	94.1	99.1	88.9	93.8
RNN	94.1	98	90	93.6
CNN + LSTM	95.2	99.5	90.8	94.9
NSL-KDD				
CNN	79.3	95.5	61.4	74.8
LSTM	75.8	95.4	54.2	69.2
GRU	79.1	95.4	61.2	74.5
RNN	76.1	88	60.5	71.7
CNN + LSTM	82.6	94.9	68.9	79.8
UNSW_NB15				
CNN	91.2	87.5	96.1	91.5
LSTM	88.9	84.8	94.8	89.5
GRU	77.9	75.3	83.2	79
RNN	71.9	65.8	91.3	76.5
CNN + LSTM	87.6	85.5	90.6	88

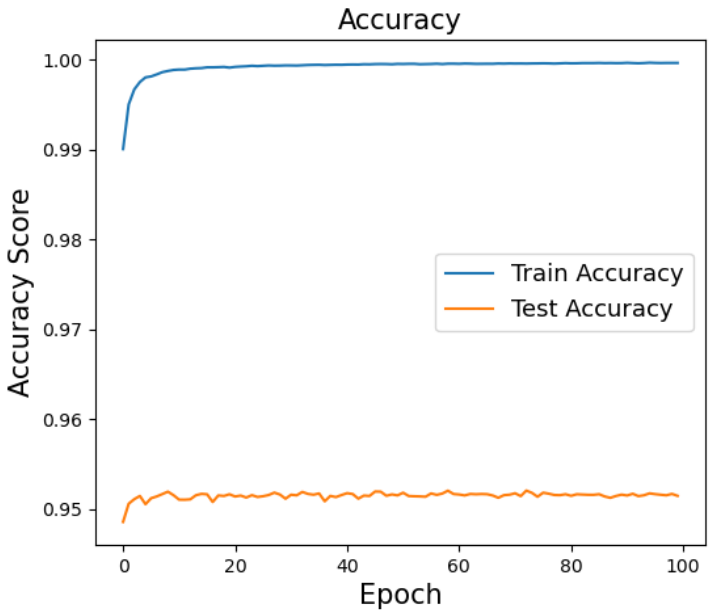

Figure 4. Training and Testing accuracy of CNN on KDD_cup99 dataset.

Our best-performed model evaluated in the KDD_cup99 dataset and detected 91% of all attack behaviors truly and 9% is wrongly detected. In case of normal behaviors, 100% of all normal behaviors are identified truly, 0.004% of them are wrongly identified. In case of the best-trained model, the evaluation of a confusion matrix is presented in Figure 5.

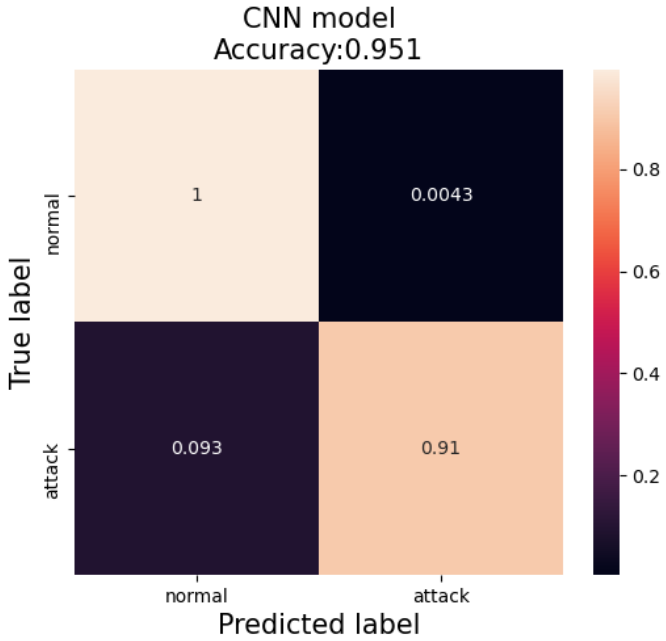

Figure 5. Confusion matrix of CNN model on balanced KDD_cup99 dataset.

6. Conclusions

In this work, we expanded and developed our previous work [1] that adapted DL models for network intrusion detection by using flow-level data. In this regard we exploited flow-level datasets such as KDD Cup99, NSL-KDD, and UNSW NB15. We applied CNN, LSTM, GRU, RNN, and hybrid CNN and LSTM DL models, the size of the batch was set to 32, the Adam optimizer with a learning rate of 0.0001 for all networks. We implemented all models by ourselves and trained on both the balanced and imbalanced data of three different datasets by using GPU CUDA acceleration. Three different datasets created header-based high-level data. This method does not depend on IP addresses or payload data. More and balanced data significantly affects DL models as shown in Figure 6.

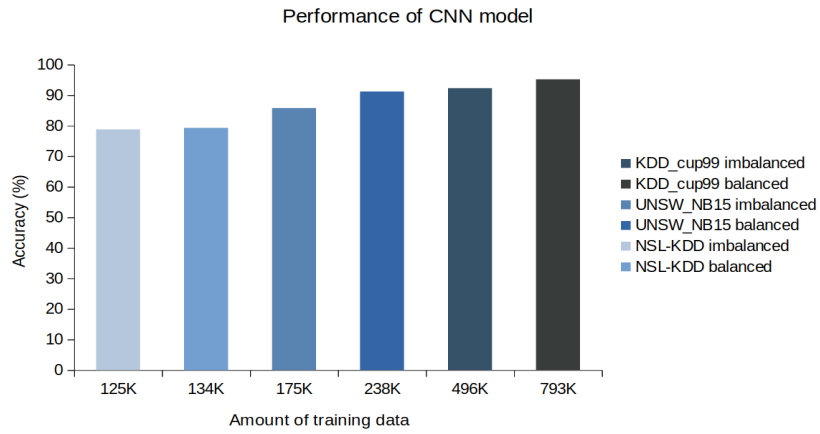

Figure 6. Impaction of data amount in case of CNN.

Tables 2 and 3 show throughput comparison of all explored models which evaluated three different balanced and imbalanced datasets. The effection of imbalanced data where an unequal set of classes of data leads to defective performance in DL models. To avoid the defective performance of models, we balanced a number of attacks and normal functions by using SMOTE over-sampling techniques.

According to the experiment, the DL models showed comparable performance, particularly CNN and combination models. Although the DL models achieved the highest accuracy score when it trained KDD_cup99 data, it is not the best solution due to this dataset is the oldest one compared to others. Figure 7 shows a comparison of the accuracy of all models on three different datasets. According to this research, we can claim that DL approaches are compatible with time-sequence data of network traffic of TCP/IP packet headers. The main benefits of the DL approach are the robustness in the achievement when providing a huge amount of data. Motivated by the current incredible performance of DL, in future work, we also aim to research cross dataset performance of IDS where training and evaluation are done on two different datasets instead of splitting the same dataset into train and test set.

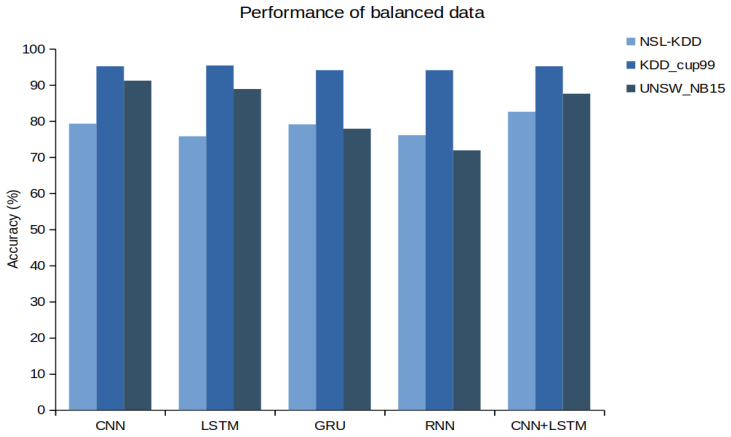

Figure 7. Results of all models using balanced data.

Author Contributions: Conceptualization, A.M. and W.K.; methodology, A.M. and J.A.; original draft preparation, A.M. and J.A.; review and editing, W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Technology development Program (no.G21S300458802) funded by the Ministry of SMEs and Startups (MSS, Korea).

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

References

1. Azizjon, M.; Jumabek, A.; Kim, W. 1D CNN based network intrusion detection with normalization on imbalanced data. In Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 19–21 February 2020; pp. 218–224.
2. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; Curran Associates Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.
3. Teyou, D.; Kamdem, G.; Ziazet, J. Convolutional Neural Network for Intrusion Detection System in Cyber Physical Systems. *arXiv* **2019**, arXiv:1905.03168.
4. Kim, J.; Kim, J.; Thu, H.L.; Kim, H. Long short term memory recurrent neural network classifier for intrusion detection. In Proceedings of the 2016 International Conference on Platform Technology and Service (PlatCon), Jeju, Korea, 15–17 February 2016; pp. 1–5.
5. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 202–206. [CrossRef]
6. Sheikhan, M.; Jadidi, Z.; Farrokhi, A. Intrusion detection using reduced-size RNN based on feature grouping. *Neural Comput. Appl.* **2021**, *21*, 1185–1190. [CrossRef]
7. Dey, R.; Salem, F.M. Gate-variants of Gated Recurrent Unit (GRU) neural networks. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; pp. 1597–1600. [CrossRef]

8. Moustafa, N.; Slay, J. UNSW-NB15: A compre- hensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015; pp. 1–6.
9. UNSW NB15. Available online: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/ (accessed on 18 January 2022).
10. Stolfo, J.; Wei, F.; Lee, W.; Prodromidis, A.; Chan, P.K. Mining with application to fraud and intrusion detection: Results from the JAM project. In Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head, SC, USA, 25–27 January 2000.
11. KDD Cup 1999. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 26 October 2007).
12. Nsl-kdd Data Set for Network-Based Intrusion Detection Systems. Available online: http://nsl.cs.unb.ca/KDD/NSLKDD.html (accessed on 18 March 2009).
13. Li, W.; Yi, P.; Wu, Y.; Pan, L.; Li, J. A new intrusion detection system based on KNN classification algorithm in wireless sensor network. *J. Electr. Comput. Eng.* **2014**, *2014*, 240217. [CrossRef]
14. Koc, L.; Mazzuchi, T.A.; Sarkani, S. A network intrusion detection system based on a Hidden Navie Bayes multiclass classifier. *Expert Syst. Appl.* **2012**, *39*, 13492–13500. [CrossRef]
15. Farnaaz, N.; Jabbar, M.A. Random forest modeling for network intrusion detection system. *Procedia Comput. Sci.* **2016**, *89*, 213–217. [CrossRef]
16. Yadav, B.; Sanjiv, T. Recent Innovations and Comparison of Deep Learning Techniques in Malware Classification: A Review. *Int. J. Inf. Secur. Sci.* **2021**, *9*, 230–247.
17. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS), New York, NY, USA, 3–5 December 2015; ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Brussels, Belgium, 2016.
18. Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A deep learning approach to network intrusion detection. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 41–50. [CrossRef]
19. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
20. Vinayakumar, R.; Soman, K.P.; Poornachandran, P. Applying convolutional neural network for network intrusion detection. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 1222–1228.
21. Wang, Y.; An J.; Huang, W. Using CNN-based representation learning method for malicious traffic identification. In Proceedings of the 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, 6–8 June 2018.
22. Hsiao, S.C.; Kao, D.Y.; Liu, Z.; Tso, R. Malware Image Classification Using One-Shot Learning with Siamese Networks. *Procedia Comput. Sci.* **2019**, *159*, 1863–1871. [CrossRef]
23. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.G.; Chen, J. Detection of malicious code variants based on deep learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
24. Alikhanov, J.; Jang, R.; Abuhamad, M.; Mohaisen, D.; Nyang, D.; Noh, Y. Investigating the Effect of Traffic Sampling on Machine Learning-Based Network Intrusion Detection Approaches. *IEEE Access* **2021**, *10*, 5801–5823. ACCESS.2021.3137318. [CrossRef]
25. Jumabek, A.; Yang, S.; Noh, Y. CatBoost-Based Network Intrusion Detection on Imbalanced CIC-IDS-2018 Dataset. *J-KICS* **2021**, *46*, 2191–2197. [CrossRef]
26. Qiu, Y.; Wu, J.; Mumtaz, S.; Li, J.; Al-Dulaimi, A.; Rodrigues, J.J. MT-MTD: Muti-Training based Moving Target Defense Trojaning Attack in Edged-AI network. In Proceedings of the ICC 2021—IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
27. Nasralla, M.M.; García-Magariño, I.; Lloret, J. Defenses against perception-layer attacks on iot smart furniture for impaired people. *IEEE Access* **2020**, *8*, 119795–119805. [CrossRef]
28. Khan, M.A.; Nasralla, M.M.; Umar, M.M.; Khan, S.; Choudhury, N. An Efficient Multilevel Probabilistic Model for Abnormal Traffic Detection in Wireless Sensor Networks. *Sensors* **2022**, *22*, 410. [CrossRef]
29. García-Magariño, I.; Nasralla, M.M.; Lloret, J. A Repository of Method Fragments for Agent-Oriented Development of Learning-Based Edge Computing Systems. *IEEE Netw.* **2021**, *35*, 156–162. [CrossRef]
30. Yu, K.; Tan, L.; Mumtaz, S.; Al-Rubaye, S.; Al-Dulaimi, A.; Bashir, A.K.; Khan, F.A. Securing critical infrastructures: Deep-Learning-Based threat in IoT. *IEEE Commun. Mag.* **2021**, *59*, 76–82 [CrossRef]
31. Lee, C.-Y.; Gallagher, P.W.; Tu, Z. Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree. *arXiv* **2015**, arXiv:1509.08985.
32. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhut-dinov, R. Dropout: A simple way to prevent neural networks from over-fitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
33. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.

34. Dunne, R.A.; Campbell, N.A. On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In Proceedings of the 8th Australian Conference on the Neural Networks, Melbourne, Australia, 11–13 June 1997; Volume 18.
35. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
36. Ketkar, N. *Deep Learning with Python*; Introduction to Keras; Apress: Berkeley, CA, USA, 2017; pp. 97–111.
37. Dillon, J.V.; Langmore, I.; Tran, D.; Brevdo, E.; Vasudevan, S.; Moore, D.; Patton, B.; Alemi, A.; Hoffman, M.; Saurous, R.A. Tensorflow Distributions. *arXiv* **2017**, arXiv:1711.10604.