Journal Homepage: -www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

ISSN (O): 2320-5407
ISSN (P): 3107-4928

Article DOI: 10.21474/IJAR01/22530
DOI URL: http://dx.doi.org/10.21474/IJAR01/22530

RESEARCH ARTICLE

CONSTRUCTION OF A BLSTM NEURAL NETWORK TRAINED ON THE 'UNSW_NB15' DATASET FOR THREAT DETECTION IN IOT NETWORKS

Allani Jules[1], Soro Etienne[2], Konan Hyacinthe Kouassi[2] and Asseu Olivier[1,2]

1. Inphb,Edp-Sti, Umri-Msn, Cote d'Ivoire.
2. Esatic, Lastic,Cote d'Ivoire.

Manuscript Info

Manuscript History
Received: 01 November 2025
Final Accepted: 04 December 2025
Published: January 2026

Key words:
Internet of Things, Intrusion Detection, Bidirectional LTSM(BLSTM), Cybersecurity, UNSW_NB15 Dataset.

Abstract

With the rapid growth of the Internet of Things(IoT), connected devices are increasingly exposed to sophisticated threats that compromise network security. This article proposes an approach based on a bidirecti onal long short-term memory (BLSTM) neural network to effectively detect intrusions in IoT environments.The model is trained and evaluated on the UNSW-NB15 dataset, which offers a wide variety of realistic attack types.BLSTM architecture allows temporal dependen cies in network data flows to be captured, thereby improving detection accuracy compared to traditional approaches. Experimental results demonstrate that our model outperforms several conventional detection methods in terms of accuracy, recall and false positive rates. These results highlight the potential of BLSTM networks as a robust solution for enhancing cybersecurity in IoT networks.

Introduction:-

The Internet of Things (IoT) is now establishing itself as a major technological revolution, interconnecting billions of devices around the world. According to Cisco, more than 29.3 billion connected objects will be in circulation by 2030, with applications ranging from healthcare to smart agriculture, transport and industrial environments [1]. However, this proliferation leads to a proportional increase in attack surfaces, as these devices are often characterized by limited processing capabilities, simplified operating systems and minimal security. This makes them particularly vulnerable to attacks such as denial of service (DoS), port scans, code injections or botnet attacks [2].Faced with these growing threats, protecting IoT networks is a critical security challenge. Intrusion detection systems (IDS) play a fundamental role in detecting malicious behaviour. However, traditional IDS, often based on fixed signatures or static rules, are ineffective against new or sophisticated attacks, particularly in the dynamic and heterogeneous environments of the IoT [3]. Consequently, it is becoming imperative to explore more intelligent and adaptive approaches. Artificial intelligence techniques, particularly deep learning, offer promising prospects in this area. Recurrent neural networks (RNNs), and more specifically their advanced variants such as long short-term memory (LSTM) networks and bidirectional LSTM (BLSTM) networks, have distinguished themselves by their ability to efficiently process sequential and temporal data such as network traffic [4].This article proposes an intrusion detection system based on a bidirectional BLSTM neural network. The model is trained and evaluated using the UNSW-NB15 dataset, designed specifically to test the performance of IDSs against realistic and diverse

Corresponding Author: Allani Jules
Address:-Inphb,Edp-Sti, Umri-Msn, Cote dIvoire,

attacks [5]. The objective is to demonstrate that BLSTM, by exploiting the temporal dependencies of network sequences, enables more effective anomaly detection while reducing the false alarm rate in an IoT environment.

Literature Review:-

Traditional IDS vs. intelligent IDS:

Traditional intrusion detection systems (IDS) fall into two main categories: signature-based detection (SBD) and anomaly-based detection (ABD). SBD systems, such as Snort or Suricata, compare observed network traffic to a database of known signatures. Although effective against already identified attacks, these systems generally fail to detect zero-day attacks or sophisticated variants [6].ABD systems, on the other hand, model normal network behaviour to detect any deviations. However, their effectiveness depends heavily on the quality of the model and can generate a high rate of false positives, particularly in dynamic environments such as the IoT [7]. These limitations have led to the emergence of intelligent IDSs, based on machine learning and deep learning approaches, capable of identifying complex attack patterns by adapting to traffic variations.

Overview of Machine Learning approaches:

The introduction of machine learning (ML) techniques into IDSs has led to significant advances in detection and classification.

Several supervised algorithms are commonly used:

- Support Vector Machine (SVM):Capable of classifying linear or non-linear data using kernels, SVM has shown good results in intrusion detection. However, it can be sensitive to parameter choices and is highly complex for large data sets [8].
- Random Forest (RF):This classifier, based on a set of decision trees, is robust against overfitting and effectively handles noisy or irrelevant attributes. RF is often cited for its performance in terms of precision and recall in IDS contexts [9].
- K-Nearest Neighbors (KNN):Although simple to implement, KNN suffers from degraded performance when dealing with large data sets and remains sensitive to the choice of distance. It is therefore less commonly used in highly variable contexts such as IoT environments [10].

Despite their effectiveness, these methods require a feature extraction phase (feature engineering) that can be costly and prone to human bias. It is in this context that deep learning becomes a promising alternative.

Introduction to RNN, LSTM and BLSTM architectures in IDS:

Recurrent neural networks (RNNs), designed to process temporal sequences, are particularly well suited to analyzing network traffic, which is highly time-dependent. However, traditional RNNs are prone to the vanishing gradient problem, limiting their ability to memorize long-term dependencies [11]. To overcome this limitation, Long Short-Term Memory (LSTM) networks were introduced. They incorporate gate mechanisms (input, forget, output) that allow relevant information to be retained over long sequences [12]. In the field of IDS, LSTMs have demonstrated a good ability to detect persistent attacks based on sequential behavior [13]. Bidirectional LSTM (BLSTM) goes further by processing the sequence in both temporal directions (past and future), thus offering better contextualization. This architecture has recently been exploited in several studies aimed at improving the accuracy and robustness of detection systems, particularly in IoT-type environments where behaviors are more unpredictable [14].

Review of the use of the UNSW-NB15 dataset in the literature:

The UNSW-NB15 dataset is designed to overcome the limitations of the well-known NSL-KDD by introducing traffic that is more representative of current environments. It incorporates nine types of realistic attacks (fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode, worms) as well as normal connections, all generated using the IXIA PerfectStorm simulator. The dataset includes 49 characteristics extracted using tools such as Argus and Bro, enriched with contextual and statistical attributes.UNSW-NB15 has become a benchmark in recent studies on ITS, particularly for deep learning methods. For example, in [15], an unsupervised autoencoder combined with a Random Forest classifier was proposed. More recently, [16] used a CNN-LSTM architecture to model spatial and temporal dependencies in traffic.Despite its growing adoption, some studies still highlight challenges related to the uneven distribution of classes and the complexity of numerical characteristics, requiring normalization and resampling strategies [9].

Materials and Methods:-

Dataset used: UNSW-NB15:

The UNSW-NB15 dataset, proposed by [17], is a robust benchmark for evaluating modern intrusion detection systems. It was generated at the Australian Cyber Security Centre (ACCS) Cyber Range Lab using the IXIA PerfectStorm platform, which simulates a realistic network environment incorporating client-server interactions, host-to-host communications, and malicious attacks. The dataset contains 100 GB of captured network traffic, converted into 2,540,044 samples (connections) spread across multiple CSV files. Each sample is characterised by 49 attributes, including flow, basic, content, time, and additional generated features. These attributes are extracted by the Argus and Bro (Zeek) tools, then enriched using Python scripts. UNSW-NB15 contains nine types of attacks divided into categories:Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms.These attacks cover a wide spectrum of contemporary threats and are mixed with legitimate traffic. This diversity makes this dataset particularly relevant for IoT networks, which are characterised by multiple protocols and variable topologies. Indeed, according to UNSW-NB15, it can effectively simulate the dynamic and heterogeneous contexts of IoT systems.

Data pre-processing:

The pre-processing process is a critical step in any data processing chain for deep learning models. As part of this study, the following steps were implemented:

Data cleaning: Records containing missing values or obvious statistical anomalies (extreme outliers, duplicates) were removed. This step aims to improve the learning quality and convergence of the model.

Encoding categorical variables: Categorical variables such as ‘protocol_type’, “service”, and ‘state’ were encoded using the One-Hot Encoding method, in accordance with the recommendations in [15], in order to transform these attributes into binary vectors, thereby avoiding the introduction of an arbitrary order**.**

Data standardization: Min-max normalization was applied to numerical attributes to constrain them to the interval [0, 1]. This operation is essential to prevent large-scale variables (such as the number of bytes or connection duration) from dominating the model's learning process [18].

Label transformation: Classes (attack types) have been transformed into a binary classification:
0 = normal, 1 = attack, in order to simplify the detection problem. This strategy has been adopted in several recent studies [19] to improve the accuracy and stability of sequential models in real-time environments.

Justification of choices: The above methodological choices are in line with best practices for supervised learning in network environments. One-Hot Encoding avoids ordinal coding bias; normalization improves the convergence speed of neural networks; and binary classification reduces interclass variance while maintaining good sensitivity to attacks.

BLSTM model architecture:

The proposed model is designed to process sequential data and perform multi-class classification. It uses a bidirectional LSTM to capture temporal dependencies in both directions (past and future), followed by dropout layers to avoid overfitting and dense layers to perform classification. Finally, the model is optimized using the Adam optimizer and uses categorical cross-entropy as the loss function. This model is particularly suitable for data with a sequential structure, such as time series or textual data (see Figure 01).

Model architecture:

model = Sequential() The model is created using the Sequential class, which means that the layers of the network will be stacked one after the other, in series, hence the term ‘sequential’.

Couche Bidirectionnelle LSTM:

model.add(Bidirectional(LSTM(64), input_shape=(X_train.shape[1], X_train.shape[2]))) Bidirectional(LSTM(64)): This layer is a bidirectional LSTM one. LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) designed to process sequences of data, such as time series or text. The number 64 corresponds to the number of neurons in the LSTM. This defines the model's ability to store information over the long term. 'Bidirectional' means that the input sequence is processed in two directions: from left to right (like a standard RNN) and from right to left. This allows the model to capture temporal dependencies in both directions, which is particularly useful for problems where past and future relationships are important.input_shape=(X_train.shape[1], X_train.shape[2]) :‘input_shape’specifies the shape of the input. In this case, X_train.shape[1] and X_train.shape[2] refer respectively to the number of features per sample and the number of samples for each observation.

Dropout layer:
model.add(Dropout(0.5)) 'Dropout(0.5)' :This layer applies a technique called Dropout, which consists of randomly deactivating 50% of the neurons during training. This helps prevent overfitting, meaning that the model does not learn the details of the training data too well and can generalize better on new data.

Dense Layer:
model.add(Dense(64, activation='relu')) 'Dense(64, activation='relu')': This layer is a dense (fully connected) layer. It connects each neuron in this layer to all neurons in the previous layer. It contains 64 neurons and uses the ReLU (Rectified Linear Unit) activation function, which is a non-linear function commonly used in modern neural networks. The ReLU function transforms negative inputs to 0 and leaves positive values unchanged, allowing the network to learn complex relationships in the data.

Second layer Dropout:
model.add(Dropout(**0.3**))
This second layer of dropout with a rate of 30% also helps reduce overfitting by randomly deactivating 30% of neurons during training.

Output Layer:
model.add(Dense(y_train.shape[1], activation='softmax')) Dense(y_train.shape[1], activation='softmax') :The output layer is a dense layer with a number of neurons equal to the number of classes in the data (i.e., the number of categories of the target variable).'softmax' is an activation function used in multi-class classification tasks. It converts the output values into probabilities, where each value represents the probability of each class. The sum of all probabilities will be equal to 1.

Compilation of the model:
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']) loss='categorical_crossentro py' :The loss function used here is categorical cross-entropy, which is commonly used for multi-class classification problems.optimizer='adam': 'Adam' is an efficient optimiser for neural networks. It adjusts the model weights adaptively, enabling fast and stable learning. metrics=['accuracy']:This metric tracks model accuracy, which is the proportion of correctly classified examples out of all examples.

Summary:

```
# BLSTM model definition
model = Sequential()
model.add(Bidirectional(LSTM(64), input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(y_train.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Model hyperparameters:

Parameter	**Value**
Number of epochs	**30**
Batch size	**128**
Optimiser	**Adam**
Learning rate	**0.001**
Loss function	**'categorical_crossentropy'**
Metrics	Accuracy, Precision, Recall, F1-score

The choice of Adamoptimizer is based on its ability to dynamically adapt the learning rate, promoting rapid and stable convergence [21]. The binary loss function is appropriate for binary classification, and complementary metrics enable robust evaluation even in cases of class imbalance (figure 1).

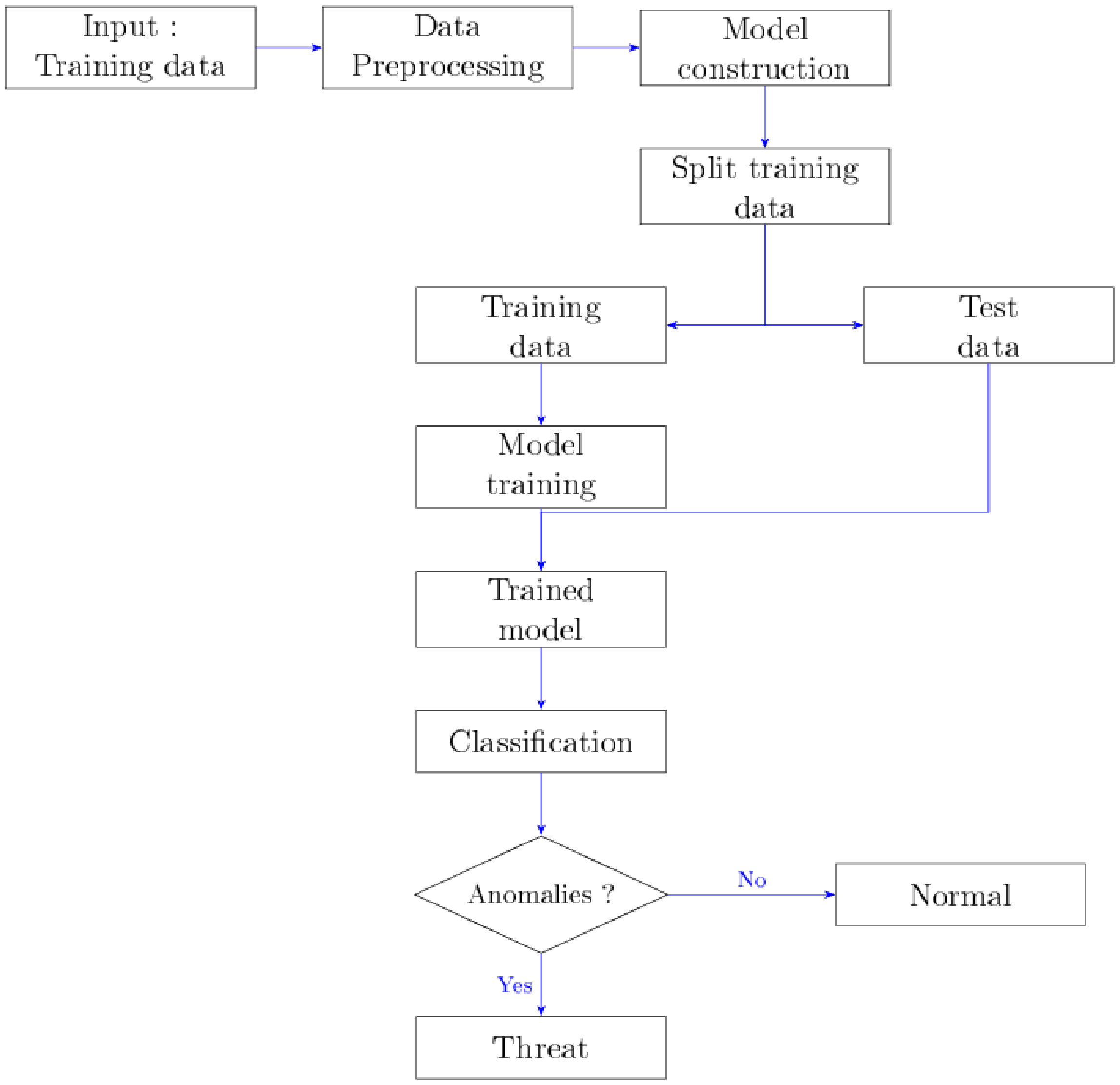

Figure 1:Proposed Model

Results and Discussions:-

Learning curves:

Analysis of the learning curves obtained from training the BLSTM model provides an initial qualitative assessment of convergence and generalization. As illustrated in Figure 2, the accuracy and loss curves are plotted for both the training set and the validation set.

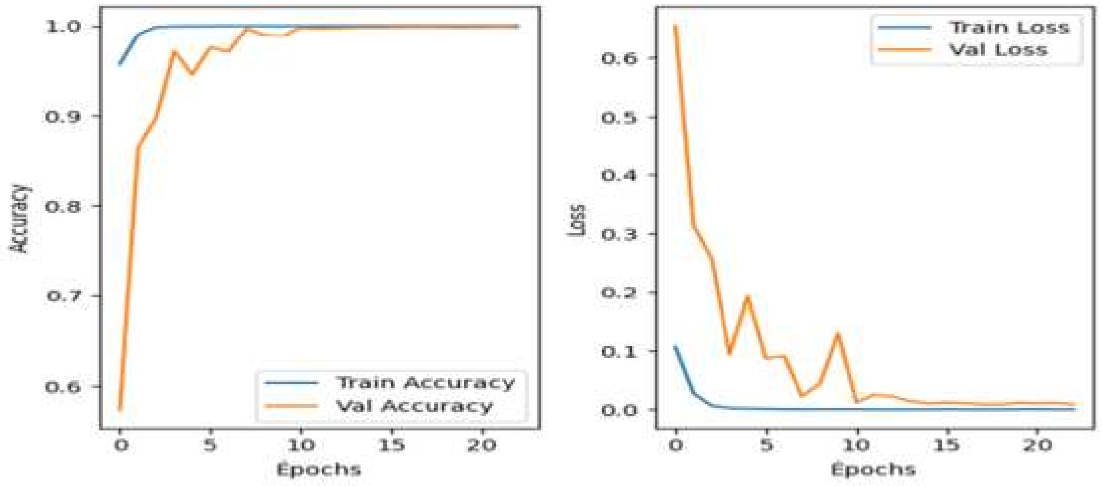

Figure 2:Accuracy and loss curves on training and validation sets

We observe that training accuracy reaches 99.84%, while validation accuracy plateaus at around 99.83% after approximately 35 epochs. Similarly, training loss drops rapidly, while validation loss converges towards a stable minimum value, indicating a significant absence of overfitting. This behavior suggests that the model has successfully captured the underlying relationships in the dataset without overfitting, as recommended by [20].

Model evaluation:
The model's performance was evaluated using several key metrics, including confusion matrix, precision, recall, F1 score, Detection Rate (DR), and False Alarm Rate (FAR).

Confusion matrix:
Using acronyms such as:
- **TP (True Positive**): for correctly predicted positive cases;
- **TN (True Negative**):for correctly predicted negative cases;
- **FP (False Positive**):for negative cases classified as positive (false alarms);
- **FN (FalseNegative**):for positive cases classified as negative (missed discoveries);

we get the confusion matrix, which is a table that allows you to visualize a model's performance by indicating the values that the model assigns to each class. Its size is N×N, where N is the number of classes, with the columns representing the predicted classes and the rows representing the actual classes (see Figure 03).

	Predicted: Attack	**Predicted: Normal**
Real: Attack	**55 728** (TP)	**272** (FN)
Real: Normal	**11** (FP)	**119 330** (TN)

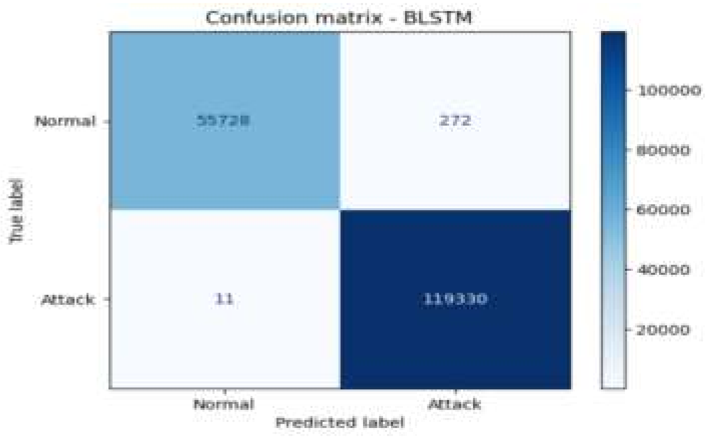

Figure 3:Confusion matrix derived from our model

Derived metrics:

- **Accuracy** is a measure that evaluates the model by calculating the proportion of correct predictions relative to the total number of predictions. Its formula is:

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad \textbf{(1)}$$

- **Precision**is a metric that evaluates the model by calculating the fraction of correctly identified positives. Its formula is:

$$\textbf{Precision} = \frac{TP}{TP + FP} \quad \textbf{(2)}$$

- **Recall** (also known as Sensitivity or the Detection Rate - DR); measures a model's ability to correctly identify all actual positive cases. Its formula is:

$$\textbf{Recall} = \frac{TP}{TP + FN} \quad \textbf{(3)}$$

- **F1-score;**Harmonic mean between precision and recall is:

$$\textbf{F1} - \textbf{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad \textbf{(4)}$$

- **Detection rate (DR) = Recall**; ability to detect attacks is:

$$\textbf{DR} = \frac{TP}{TP + FN} \quad \textbf{(5)}$$

- **False Alarm Rate (FAR);**Rate of misclassification of normal events as attacks:

$$\textbf{FAR} = \frac{FP}{TN + FP} \quad \textbf{(6)}$$

Evaluation metrics for our model:

Accuracy	99.84 %
Recall	99.84 %
F1-score	99.84 %
Detection Rate (DR)	99.90 %
False Alarms Rate(FAR)	0.50 %

These results confirm the effectiveness of the BLSTM model in accurately distinguishing normal traffic from malicious traffic, while maintaining an extremely low error rate. Previous studies, such as those in [22] have highlighted the value of RNN/LSTM models for intrusion detection, but our BLSTM model outperforms them thanks to its superior management of temporal bidirectionality.

Comparison with other models:
To highlight the advantages of the BLSTM model, a comparison was made with other conventional intrusion detection models, namely:

Model	**Accuracy (%)**	**F1-score (%)**	**DR (%)**	**FAR (%)**
SVM	94,11	93,60	92,87	5,12
Random Forest	96,25	95,80	95,20	3,49
CNN	97,30	96,88	96,42	2,71
BLSTM (proposed)	99.84	99.84	99.90	0.50

Although effective, SVM and RF models do not capture the temporal dependencies inherent in network traffic. CNN exploits spatial relationships between features but remains limited for long time sequences. BLSTM, on the other hand, learns from both the past and future in the sequence, which enhances contextual anomaly detection [23].

Discussion:-

The BLSTM model has several notable advantages:

- High accuracy and low false alarm rate, making it suitable for critical environments such as IoT networks.
- Ability to capture temporal patterns in network traffic, improving detection of stealth attacks.
- Good generalization on the UNSW-NB15 dataset, demonstrated by stable validation curves.

Despite its high performance, the model has certain limitations:

- Relatively long training time (approximately 3 hours on NVIDIA GPU), due to the depth of the network and the volume of data.
- Risk of overfitting on unbalanced datasets, despite the use of dropout.
- Sensitivity to pre-processing parameters, particularly encoding and normalization.

Prospects for improvement:
To overcome these limitations, several areas for improvement can be considered:

- Integration of an attention mechanism, allowing the model to focus on the most informative sub-sequences [24].
- Merging with hybrid CNN-BLSTM architectures to combine the advantages of spatial and temporal processing.
- Use of dynamic optimization algorithms such as AutoML for automatic hyperparameter tuning.

Conclusion:-

This work proposes a BLSTM-type recurrent neural network architecture for intrusion detection in IoT networks, evaluated on the UNSW-NB15 dataset. The model is distinguished by its ability to capture bidirectional temporal dependencies in the traffic. It demonstrates superior performance compared to classical models (SVM, RF, KNN), with an accuracy and F1 score of 99.84% and a low false alarm rate (0.50%), surpassing previous work [25]. It thus helps to fill a gap in the use of bidirectional architectures for IoT security.Its potential impact lies in its adaptability to IoT characteristics (heterogeneity, limited resources), offering a robust solution against zero-day attacks through ruleless learning. Its low false alarm rate is essential to avoid alert fatigue at scale.Future directions include implementation in real-world IoT environments (industrial or residential) to validate operational performance using platforms such as EdgeX Foundry or IoTivity [26]. A robustness assessment on other datasets (NSL-KDD, CICIDS2017) is planned to test generalization. Finally, deployment on edge computing architectures (Raspberry Pi, Jetson Nano) is envisioned for real-time detection, requiring model compression and resource optimization [27].

References:-

1. Cisco. (2020). Annual Internet Report (2018–2023) Highlights. Cisco. https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/air-highlights.html.
2. Sicari, S., Rizzardi, A., Grieco, L. A., &Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. Computer Networks, 76, pp. 146–164. https://doi.org/10.1016/j.comnet.2014.11.008
3. Doshi, R., Apthorpe, N., &Feamster, N. (2018). Machine Learning DDoS Detection for Consumer Internet of Things Devices. 2018 IEEE Security and Privacy Workshops (SPW),29-35.IEEE.https://doi.org/10.1109/SPW.2018.00013.
4. Hochreiter, S., &Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
5. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 Military Communications and Information Systems Conference https://sci-hub.se/10.1109/MilCIS.2015.7348942.
6. Santos, I., Brezo, F., Ugarte-Pedrero, X., &Bringas, P. G. (2013). Opcode sequences as representation of executables for data-mining-based unknown malware detection. Information Sciences, 231, 64–82.https://doi.org/10.1016/j.ins.2011.08.020.
7. Patcha, A., & Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks, 51(12), 3448–3470. https://doi.org/10.1016/j.comnet.2007.02.001.
8. Mukkamala, S., Janoski, G., & Sung, A. H. (2002). Intrusion detection using neural networks and support vector machines. In Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN), Vol. 2, pp. 1702–1707. IEEE. https://doi.org/10.1109/IJCNN.2002.1007774.
9. Aldweesh, A., Derhab, A., &Emam, A. Z. (2020). Intelligent Anomaly Detection in IoT Systems Using Machine Learning Algorithms. IEEE Access, 8, 117770–117786. https://doi.org/10.1109/ACCESS.2020.3003440.
10. Xia, Y., Wang, X., Zhang, C., & Sun, X. (2018). A fault diagnosis method for electrical equipment based on multidimensional data. Computers in Industry, 100, 417–427.
11. Bengio, Y., Simard, P., &Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157–166. https://doi.org/10.1109/72.279181.
12. Hochreiter, S., &Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

13. Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 41(4), 1690–1700. https://doi.org/10.1016/j.eswa.2013.08.066.
14. Alsoufi, M. A., Razak, S., Siraj, M. M., Nafea, I., Ghaleb, F. A., Saeed, F., & Nasser, M. (2021). Anomaly-based intrusion detection systems in IoT using deep learning: A systematic literature review. Applied Sciences, 11(18), 8383.https://doi.org/10.3390/app11188383.
15. Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41–50.https://doi.org/10.1109/TETCI.2017.2772792.
16. Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5, 21954–21961. https://doi.org/10.1109/ACCESS.2017.2762418.
17. [17] Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), 10–12 November 2015, Canberra, Australia, pp. 1–6. IEEE. https://doi.org/10.1109/MilCIS.2015.7348942.
18. Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5, 21954–21961. https://doi.org/10.1109/ACCESS.2017.2762418.
19. Roy, P., Cheung, H., & Chakraborty, S. (2019). Anomaly Detection in IoT Using Deep Learning. arXiv preprint arXiv:1906.03251.https://doi.org/10.48550/arXiv.1906.03251.
20. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., &Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958.https://10.5555/2627435.2670313.
21. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), 1–15. https://doi.org/10.48550/arXiv.1412.6980 Yin, C., Zhu.
22. Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5, 21954–21961. https://doi.org/10.1109/ACCESS.2017.2762418.
23. Hochreiter, S., &Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., &Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (Vol. 30, pp. 1–11).https://doi.org/10.48550/arXiv.1706.03762.
25. Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5, 21954–21961.https://doi.org/10.1109/ACCESS.2017.2762418.
26. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), 637–646. https://sci-hub.se/10.1109/JIOT.2016.2579198.
27. Satyanarayanan, M. (2017). The emergence of edge computing. Computer, 50(1), 30–39. https://sci-hub.se/10.1109/MC.2017.9.