

XMeDNN: An Explainable Deep Neural Network System for Intrusion Detection in Internet of Medical Things

Mohammed M. Alani^{1,2}^a, Atefeh Mashatan³^b and Ali Miri¹^c

¹*Department of Computer Science, Toronto Metropolitan University, Toronto, Canada*

²*School of IT Administration and Security, Seneca College, Toronto, Canada*

³*Ted Rogers School of Information Technology Management, Toronto Metropolitan University, Toronto, Canada*

Keywords: IoMT, IoT, DNN, Deep Neural Network, Intrusion Detection.

Abstract: The rapid growth in the adoption of Internet of Things in various areas of our daily lives makes it an interesting target to malicious actors. One area that is witnessing accelerating growth in smart device adoption is health and medical services. The growth in Internet of Medical Things is triggering increased interest in privacy, confidentiality, and intrusion detection. In this paper, we present a deep neural network designed to detect attacks on Internet-of-Things devices in medical settings. The proposed system was tested using WUSTL-EHMS-2020 dataset. Tests showed that the proposed deep learning system can deliver excellent performance with accuracy 97.578%, and a false-positive rate of 3.12%.

1 INTRODUCTION

As the world witnesses rapid growth in Internet-of-Things (IoT) adoption, medical and health applications show an even more rapid growth. According to (Statista, 2022a), electronic health devices expected to hit a revenue of US\$12.97bn before the end of 2022. That rapid growth makes these devices an interesting target for malicious actors. Figure 1 shows the revenue of the eHealth Device market from 2017, with projections until 2027. As shown in the figure, the expected market growth is about 50% in the coming five years. This significant growth comes with noticeable increase in the attacks conducted on such devices.

Internet-of-Medical-Things (IoMT), in addition to the security threats in common with other IoT devices, pose a particularly significant privacy risk as such devices handle extremely sensitive private information. While some attacks, such as false-data injection attacks, might be categorized as a low-to-medium risk in other IoT contexts, they can be life-threatening in IoMT context. When data emanating from a smart-home based thermometer is maliciously falsified, it can cause the air-conditioning system to operate in an

undesired manner that could cause discomfort. However, when falsified data is maliciously injected to appear as originating from a thermometer attached to a patient, it can cause the smart treatment system to miscalculate the medication dosage given to a patient, and might cause the loss of life.

Man-In-The-Middle (MITM) attacks are of particular concern in IoMT context. In these attacks the attacker operates in the middle between an IoMT device and a server, or between IoMT devices. Techniques that the attacker might use would include spoofing. In spoofing attacks, the attacker impersonates a legitimate entity such as a server or an IoMT device. When impersonating a server, the attacker would intercept the sensor's traffic, and would be able to send falsified commands to the IoMT device(Alani, 2014). On the other hand, when the attacker impersonates an IoMT device, it could intercept commands issued from the server and prevent the actual IoMT device from doing its job, or send false information to the server to sabotage the collected data at the server's side. In both scenarios, the outcome of a successful attack could be of catastrophic consequences.

In this paper, we present an Intrusion Detection System (IDS) designed to capture different types of attacks on IoMT environments using features extracted from network-flows. The proposed system is based on explainable Deep Neural Network (DNN)

^a <https://orcid.org/0000-0002-4324-1774>

^b <https://orcid.org/0000-0001-9448-9123>

^c <https://orcid.org/0000-0001-5991-7188>

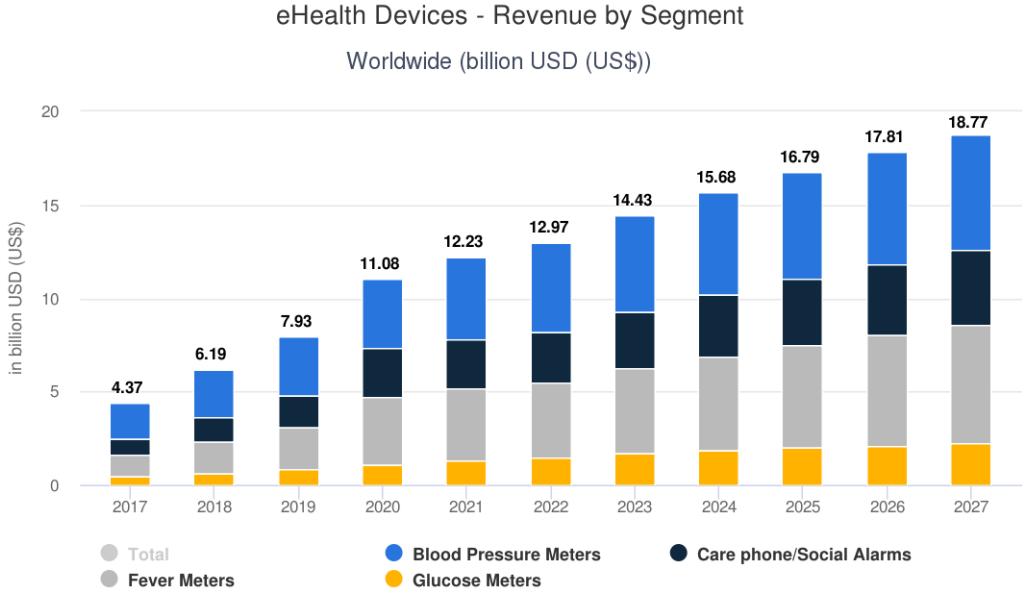


Figure 1: Historical eHealth Device Market Revenue by Segment with Future Projections (Statista, 2022a).

that processes the information extracted from the network to make predictions to help detect and block malicious traffic. The proposed DNN solution is explained using Shapley Additive Explanation (SHAP) to identify how the selected features impact the classifier’s decision.

The following section reviews relevant literature, while Section 3 presents an overview of the proposed system. The details of the used dataset and its pre-processing is explained in Section 4, along with the implementation environment. Section 5 outlines the implemented experiments and their results, followed by a thorough discussion of the model’s explainability in Section 6. The presented results are discussed with comparative analysis in Section 7, followed by the last section presenting conclusions and future work.

2 RELATED WORKS

Intrusion detection has been one of the active areas of machine-learning research for a long time. General IoT IDS research has also gained a lot of momentum in the past few years with applications ranging from classical machine learning (Alani, 2022) to deep learning (Ahmad et al., 2021), to federated learning (Alazab et al., 2021). This was enabled by the availability of many realistic and synthetic datasets that captured traffic such as Bot-IoT (Koroniots et al., 2019), TON_IoT (Moustafa, 2021), and IoT-ID (Kang et al., 2019).

The main obstacle faced by the such research within the medical context of IoT was the lack of

realistic datasets that captured attacks and normal traffic data. Many research papers designed machine learning-based intrusion detection systems using datasets that are do not capture medical IoT traffic. This research ignores the fact that traffic generated and received by medical IoT devices has unique characteristics such as the size and number of packets, the amount of data generated, and the sensitivity of the data carried on that traffic. However, the dataset used in our research, WUSTL-EHMS-2020 captures IoT traffic within medical context. In this section we will review research conducted within the IoMT context specifically.

Hady et al. presented the WUSTL-EHMS-2020 dataset in (Hady et al., 2020) along with a proposed intrusion detection system based on machine learning. The proposed system tested different classifier algorithms, with a k-Nearest Neighbor (kNN) model outperforming other classifiers with an accuracy of 91.56%, and AUC of 0.8748. The proposed classifiers suffered from low AUC generally due to ignoring the class imbalance issue in the dataset.

In 2021, Lee et al. proposed a multi-class classification-based IDS for IoT in medical context named M-IDM (Lee et al., 2021). The proposed system employed Convolutional Neural Networks (CNN) to classify traffic type in a multi-class fashion. The proposed architecture classifies the data into multiple classes: Critical, informal, major, and minor, for intrusion detection. The proposed system used a non-public dataset collected from actual IoMT devices over a period of time. When tested, the proposed system delivered an AUC of 0.967 and an F_1

score of 0.937, with an accuracy of approximately 94%.

In 2022, Gupta et al. presented an IoMT network IDS system based on a tree classifier (Gupta et al., 2022). The proposed system utilized WUSTL-EHMS-2020 dataset to build a tree classifier-based intrusion detector that focuses on dimensionality reduction to speed up the detection process. The proposed system also utilized GridSearchCV to find the best estimator in a group of different random forest classifiers. The proposed system was tested, and performed with an accuracy of 94.23% with an F_1 score of 0.938.

Wahab et al. presented, in 2022, a machine learning-based intrusion detection system for IoMT devices (Wahab et al., 2022). The proposed system utilizes a hybrid model consisting of long short-term memory (LSTM) and gated recurrent unit (GRU) to create a highly accurate classifier. The issue with this research is that it was trained and tested using CI-CDDoS2019 dataset (Sharafaldin et al., 2019). This dataset captures only Distributed Denial of Service (DDoS) attacks on computers. It is not an IoT dataset, nor is it a IoMT dataset. Despite the high accuracy of 99.01%, this research does not actually address the IoMT security issues.

Nandy et al. proposed, in 2022, an IoMT intrusion detection system based neural networks optimized by swarm intelligence (Nandy et al., 2022). The proposed system falls into the same issue mentioned about the previous research article; it trains and tests the classifier using a non-IoMT dataset. The dataset used in the proposed system was TON_IoT. The dataset, although captured from IoT devices, it did not contain any IoMT traffic. Hence, the produced high accuracy of 99.5% is not representative of the quality of detection for IoMT traffic.

Further information can be found in (Si-Ahmed et al., 2022) about other directions of research that addresses IoMT security using machine-learning.

3 PROPOSED SYSTEM

In this paper, we proposed a DNN-based intrusion detection system that is trained to detect attack in IoMT environments, named XMeDNN. Figure 2 shows an overview of the proposed system.

As shown in the figure, the training stage includes preprocessing, and hyperparameter tuning to reach the best possible trained DNN. Once the training stage is completed, the trained DNN model is then sent to the deployment stage.

At the deployment stage, the network traffic is captured as raw packets. These raw packets are

then passed through a network-flow analysis package named ARGUS (Statista, 2022b). This tool, extracts network-flow features from raw captured traffic. These features are then passed as the input to the trained DNN model for inference. The trained DNN model then produces a prediction whether the captured network flow is normal or attack traffic. This prediction can later be linked a firewall to block the source IP address for attack traffic.

4 THE DATASET

The dataset used to train and test the proposed model, WUSTL-EHMS-2020, was introduced in (Hadji et al., 2020). The dataset was created using actual IoMT devices connected in a testbed setup. The devices used to build the dataset were SpO_2 measurement device, blood pressure measurement device, Electro-Cardio-Graph (ECG) device, and a thermometer. The data collected included network traffic in addition to device readings. The traffic files were processed using a tool named ARGUS, that produces network-flow features from raw captured packets.

The original dataset was composed of 16,138 samples (14,272 normal, and 2,046 attack). Each sample contains 35 network-flow features, and eight device reading features. As our proposed model was focused on detecting attacks through the analysis of network traffic, we removed the device biometric readings.

4.1 Implementation Environment

The data preprocessing, model training, model testing, and explainability was performed in an environment with the following specifications:

- Processor: AMD Ryzen 5 3600 (4.2GHz)
- RAM: 128GB
- GPU: Nvidia RTX3060Ti
- Nvidia CUDA v11.3
- Operating System: Windows 10 Pro
- Python v3.10.0
- TensorFlow v2.9.1
- Sci-Kit Learn v1.1.3
- Matplotlib v3.5.3
- SHAP v0.39.0

4.2 Preprocessing

Upon close examination of the dataset, we made the following observations:

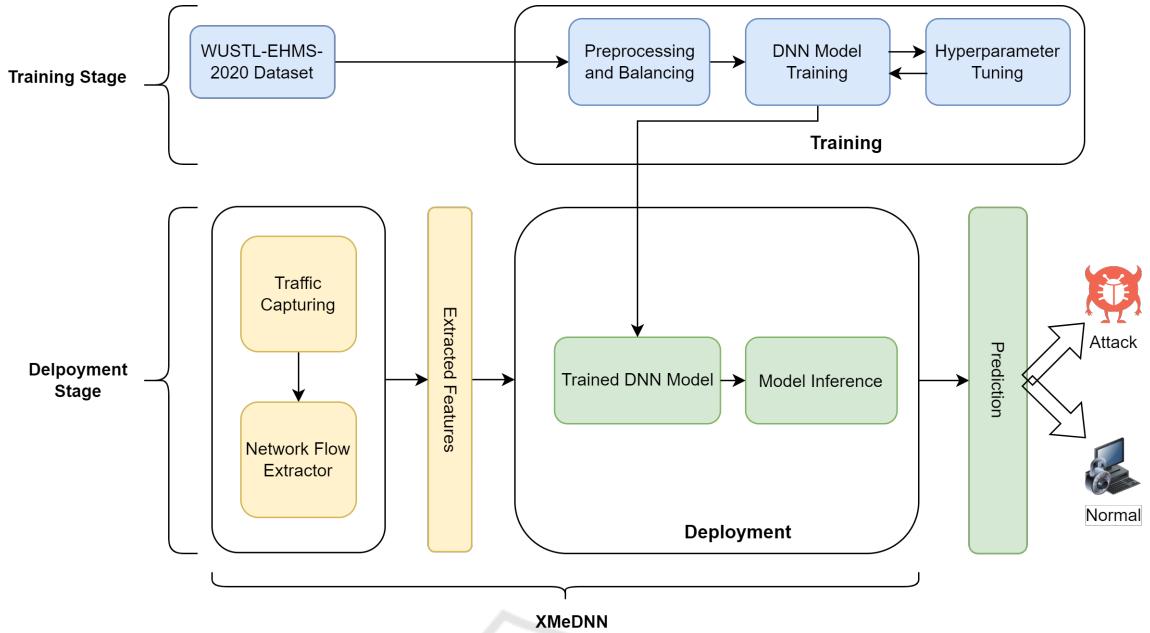


Figure 2: An overview of the proposed system XMeDNN.

1. There is a noticeable imbalance between the “normal” class and the “attack” class, with the attack class making up only 12.5% of the total number of samples.
2. The features included in the dataset include many host specific features such as source and destination IP addresses, and source and destination MAC addresses.
3. The dataset includes two samples with invalid data (in the form of text in features that are supposed to hold numbers).

Based on these observations, the preprocessing steps shown in Algorithm 1 was performed.

Algorithm 1: Dataset Preprocessing.

Input: Original dataset
Output: Preprocessed dataset

```

Array ← Dataset
Array ← Array(remove_unwanted_features)
for i in 16,318 do
    if Array(i) = invalid_data then
        Remove_Sample Array(i)
    end
end
Array ← RandomOverSampling(Array)
Dataset ← Array

```

The first step in the preprocessing of the dataset was to remove unwanted features. The features removed were host-specific, such as IP and MAC ad-

dresses. If these features were to be used in training, they would cause poor classifier generalization as the model would be trained to capture traffic from a specific source as the malicious source, and would perform poorly when tested with attacks from other sources. This resulted in the removal of seven features, with 28 features remaining.

The next step in the preprocessing, as shown in the algorithm, was to detect and remove samples with invalid data. There were only two samples that contained invalid data. This leaves 16,316 samples (14,270 normal, and 2,046 attack).

The last step was to address the noticeable imbalance between the normal and attack samples. This was addressed through random over-sampling of the minority class (attack) until the two classes have an equal number of samples. This produced a dataset with 28,540 samples (14,270 normal, and 14,270 attack), with 28 features.

According to (Raschka et al., 2022), it is a common best practice to normalize the data used in training and testing neural networks. This feature-wise normalization is said to help neural networks perform better and avoid being impacted by the wide range of different input features. As our proposed system was based on DNN, we proceeded to normalize the data using MinMaxScaler from the Sci-Kit Learn (Learn, 2022) package in Python.

5 EXPERIMENTS AND RESULTS

The data was randomly split into 75% training subset, and 25% testing subset. The random split took into account maintaining the balance between the normal and attack classes.

5.1 Experiment Setup

To ensure that the best possible performance can be reached using our proposed DNN, we used a hyperparameter optimization package named Talos (autonomio, 2022). This package performs a thorough search in different ranges of hyperparameters that we chose to fine-tune the neural network. Utilizing this optimization package, along with our experience, the following hyperparameters were capable of delivering the best performance for our proposed system:

- Number of input neurons: 28
- Number of hidden layers: 3
- Number of neurons in hidden layers: 64-32-8
- Activation functions in hidden layers: elu
- Number of neurons in the output layer: 1
- Activation function in the output layer: sigmoid
- Initializer: He Normal Initializer
- Optimizer: Nadam with learning rate of 0.01
- Loss function: Binary cross-entropy
- Epochs: 150
- Batch size: 20

To prove DNN's superiority over classical machine learning in solving this classification task, we set up the experiments to train and test five different types of classifiers:

1. DNN Multi-Layer Perceptron (MLP)
2. Logistic Regression (LR)
3. Decision Tree (DT)
4. Gaussian Naive Bayes (GNB)
5. eXtreme Gradient Boosting (XGB)

Each of these classifiers were trained using the training subset, and tested with the testing subset. The results of testing can be found in Table 1.

Table 1: Testing Results for the Five Classifiers.

	Accuracy	Precision	Recall	F_1 Score
DNN	0.96878	0.96995	0.96887	0.96877
LR	0.68053	0.70065	0.68001	0.67192
DT	0.96114	0.96337	0.96125	0.96110
GNB	0.69200	0.73657	0.69128	0.67627
XGB	0.92239	0.92342	0.92247	0.92235

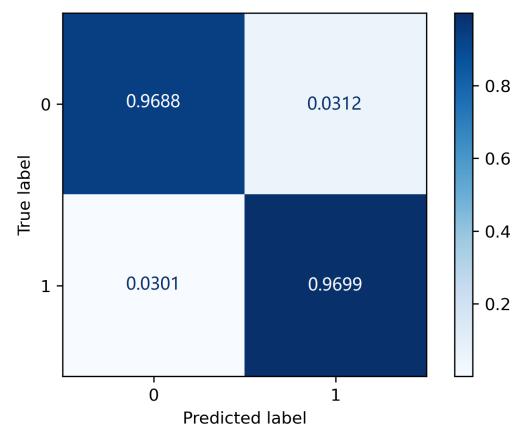


Figure 3: Confusion Matrix Plot for the DNN Classifier.

As we examine the results in the table, we can see that the proposed DNN architecture slightly outperforms DT classifier with F_1 score of 0.968, and they both noticeably outperform the other classifiers. Figure 3 shows the confusion matrix plot of the proposed DNN classifier. The confusion matrix plot shows that the proposed classifier achieves a false-positive rate of 3.12%, and a false-negative rate of 3.01%.

Figure 4 shows the changes in the loss and accuracy values as the training proceeds until 150 epochs. This figure shows that the accuracy was improving while the loss is gradually being reduced as the training proceeds.

5.2 10-Fold Cross-Validation

To ensure that the proposed classifier can generalize well, we performed 10-fold cross-validation. In a 10-fold cross-validation, the dataset is randomly divided into 10 different subsets. Ten training cycles are performed where one of the subsets is used for testing and the remaining nine are used for training. At the end of the 10 cycles, each one of the 10 subsets was used for testing, while excluded from training, once. At the end of the cross-validation process, if the results of the 10 folds have low standard deviation, the mean value of the performance metrics can be considered a reliable result. The results of applying 10-fold cross validation on our proposed model can be found in Table 2.

By examining the results shown in the table, it is clear that the standard deviation is minimal, and the results of the 10 folds are very aligned with each other, with an average accuracy of 97.57%, and an average F_1 score of 0.9763. This result establishes that the proposed classifier generalizes well beyond its training dataset.

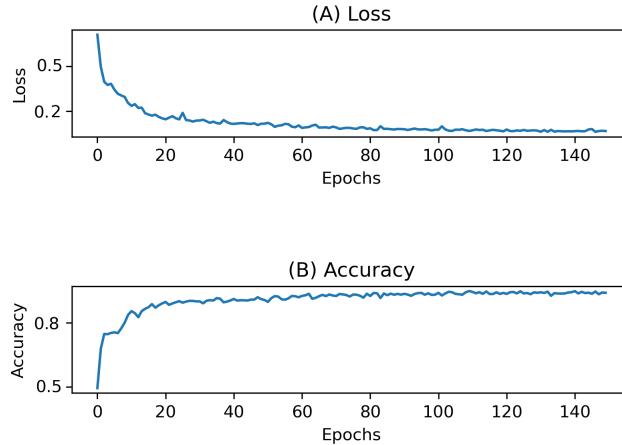


Figure 4: (A) The change in loss value with epochs.
(B) The change in accuracy with epochs.

Table 2: Results of the 10-fold cross-validation.

Fold	Accuracy	Precision	Recall	F_1 Score
1	0.98072	0.96390	0.99930	0.98128
2	0.97617	0.95363	0.99927	0.97592
3	0.97582	0.95411	0.99929	0.97618
4	0.97196	0.94870	0.99789	0.97267
5	0.97687	0.95588	1.00000	0.97744
6	0.97512	0.95133	1.00000	0.97506
7	0.97267	0.95006	1.00000	0.97439
8	0.97582	0.95396	1.00000	0.97644
9	0.97547	0.95385	1.00000	0.97638
10	0.97722	0.95766	0.99859	0.97770
Mean	0.97578	0.95431	0.99943	0.97634
STDev	0.00229	0.00406	0.00068	0.00215

6 MODEL EXPLAINABILITY

Model's explainability increases trust in the trained model by explaining how the prediction decisions in the classifier are made. When the impact of different features on the prediction decision is explained, the neural network does not remain a black-box system where decisions are made without proper understanding of the impact of features' values on the decision process.

For our system, we chose the use of SHAP values for explanation of the proposed model. SHAP values are model-agnostic explanation method that is based on game theory. It was first introduced in (Lundberg and Lee, 2017) in 2017. The operation of SHAP involves calculating the impact of each feature by calculating the difference between the model's performance with the feature, and without the feature. This helps in understanding the individual contribution of the feature to the prediction. The explainer type used in our experiment was DeepExplainer.

Figure 5 shows the SHAP values summary plot of the top five features with the highest importance. These five features are ordered in descending order from the feature with the highest impact on the deci-

sion to the lowest.

In the figure, the dots shown on the left side of the axis are the values that drag the prediction value down, which makes the prediction closer to "normal" traffic, while the dots on the right side of the axis push the prediction value up which makes the prediction closer to "attack" traffic. The dots in red represent a high value of the feature, while the blue dots represent a low value of the feature.

As shown in Figure 5, the feature with the highest impact is SrcLoad. This feature measures the data rate in bits-per-second for the data originating from the source host within a network flow. The figure shows that lower values of SrcLoad push the prediction closer toward an "attack", while higher values bring the prediction closer to "normal" traffic. This is consistent with the fact that the attacker would usually utilize low data rates to avoid detection, while normal traffic, as shown in the figure, can have varying high rates. This phenomenon also explains the fourth feature, Load, which captures the average load on both sides in bits-per-second.

The feature with the second and third features in terms of impact, as shown in the figure are DintPkt and DstJitter. DintPkt feature captures the average inter-packet arrival time at the destination side in the network flow, while DstJitter captures the average jitter at the destination side. The figure shows that low values, and extremely high values or inter-packet timing are indicative of normal traffic. On the other hand, high values are indicative of an attack. Logically, this is aligned with the jitter value that captures the variation of the arrival time of the packet from true periodicity. Both of these features explanation is consistent with the fact that attack packets are mostly manually crafted by the attacker and would not be sent in an organized flow rate.

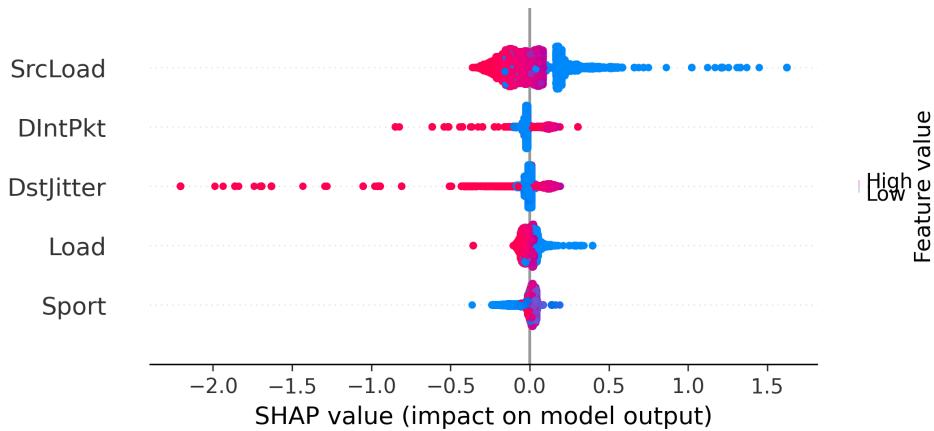


Figure 5: SHAP Summary Plot for the Top Five Features.

Table 3: Comparison of the results of the proposed system with related works.

Research	Dataset	Classifier	Accuracy	F_1 Score	FPR	Explainable
Gupta et al. (Gupta et al., 2022)	WUSTL-EHMS-2020	Tree	94.23%	0.938	6%	x
Hadly et al. (Hadly et al., 2020)	WUSTL-EHMS-2020	kNN	91.56%	-	-	x
Lee et al. (Lee et al., 2021)	(Lee et al., 2021)	CNN	94%	0.937	5.3%	x
Proposed system	WUSTL-EHMS-2020	MLP	97.57%	0.976	3.1%	✓

The fifth feature in the figure is Sport, which captures the source port number. Figure 5 shows that low port numbers are an indicator of attack, while medium to high port numbers are seen as an indicator of normal traffic. This is highly aligned with the fact that in normal traffic, the source port number is dynamically assigned with the range of the ephemeral ports. Although the IETF standard shows 1,024-65,536 as a usable range (Reynolds, 1992), most operating system utilize high ranges of ports for dynamic usage. Many Linux distributions use the range of 32,768-61,000, while recent Windows versions use the range 49,152-65,535 (Techopedia, 2014). This is also aligned with the fact that attack packets are manually crafted by the attacker, and hence do not always adhere to the port selection industry standards.

7 DISCUSSION

Testing results obtained in the initial testing were validated by the results obtained by the 10-fold cross-validation process. This proves that the trained classifier can generalize well beyond its training dataset.

Table 3 shows a comparison of the results obtained with relevant previous works. We excluded the papers that did not use IoMT dataset as their results are not valid in the IoMT context.

As shown in the table, our proposed system outperforms all related works by a good margin in terms of accuracy, F_1 score, as well as the false-positive rate.

As clearly stated in Table 3, none of the related works utilized explainable machine learning to ex-

plain the impact of different features on the classifier's decision.

Model's explainability of the top five features, presented in Section 6, is noticed to be highly aligned with general cybersecurity experience in the conducted attacks. This further validates the accuracy of the results obtained.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a DNN-based intrusion detection system for IoMT environments. The proposed system was trained and tested using WUSTL-EHMS-2020 dataset. The proposed system, namely XMeDNN, delivered an average accuracy of 97.578% in 10-fold cross-validation, with an average F_1 score of 0.97634. The proposed system delivered better performance metrics when compared to related previous works, as discussed in Section 7.

Future directions of this research include the following:

1. Improving the scope of attacks covered by the proposed IDS by exploring additional datasets.
2. Performing further validation by testing with different datasets.
3. Exploring the use of federated learning to maintain data-source privacy while improving the system's accuracy.

REFERENCES

- Ahmad, Z., Shahid Khan, A., Nisar, K., Haider, I., Hassan, R., Haque, M. R., Tarmizi, S., and Rodrigues, J. J. (2021). Anomaly detection using deep neural network for iot architecture. *Applied Sciences*, 11(15):7050.
- Alani, M. M. (2014). *TCP/IP Model*, pages 19–50. Springer International Publishing, Cham.
- Alani, M. M. (2022). Iotprotect: A machine-learning based iot intrusion detection system. In *2022 6th International Conference on Cryptography, Security and Privacy (CSP)*, pages 61–65. IEEE.
- Alazab, M., RM, S. P., Parimala, M., Maddikunta, P. K. R., Gadekallu, T. R., and Pham, Q.-V. (2021). Federated learning for cybersecurity: Concepts, challenges, and future directions. *IEEE Transactions on Industrial Informatics*, 18(5):3501–3509.
- autonomio (2022). talos. [Online; accessed 22. Dec. 2022].
- Gupta, K., Sharma, D. K., Datta Gupta, K., and Kumar, A. (2022). A tree classifier based network intrusion detection model for internet of medical things. *Computers and Electrical Engineering*, 102:108158.
- Hady, A. A., Ghubaish, A., Salman, T., Unal, D., and Jain, R. (2020). Intrusion detection system for healthcare systems using medical and network data: A comparison study. *IEEE Access*, 8:106576–106584.
- Kang, H., Ahn, D. H., Lee, G. M., Yoo, J. D., Park, K. H., and Kim, H. K. (2019). Iot network intrusion dataset. *IEEE Dataport*.
- Koroniotsis, N., Moustafa, N., Sitnikova, E., and Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796.
- Learn, S.-K. (2022). scikit-learn: machine learning in Python — scikit-learn 1.2.0 documentation. [Online; accessed 22. Dec. 2022].
- Lee, J. D., Cha, H. S., Rathore, S., and Park, J. H. (2021). M-idm: A multi-classification based intrusion detection model in healthcare iot. *Computers, Materials and Continua*, 67(2):1537–1553.
- Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.
- Moustafa, N. (2021). A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. *Sustainable Cities and Society*, 72:102994.
- Nandy, S., Adhikari, M., Khan, M. A., Menon, V. G., and Verma, S. (2022). An intrusion detection mechanism for secured iomt framework based on swarm-neural network. *IEEE Journal of Biomedical and Health Informatics*, 26(5):1969–1976.
- Raschka, S., Liu, Y. H., Mirjalili, V., and Dzhulgakov, D. (2022). *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing Ltd.
- Reynolds, J. (1992). RFC 1340: Assigned Numbers. [Online; accessed 22. Dec. 2022].
- Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. (2019). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCS)*, pages 1–8. IEEE.
- Si-Ahmed, A., Al-Garadi, M. A., and Boustia, N. (2022). Survey of machine learning based intrusion detection methods for internet of medical things. *arXiv preprint arXiv:2202.09657*.
- Statista (2022a). eHealth Devices - Worldwide | Statista Market Forecast. [Online; accessed 22. Dec. 2022].
- Statista (2022b). eHealth Devices - Worldwide | Statista Market Forecast. [Online; accessed 22. Dec. 2022].
- Techopedia (2014). What is an Ephemeral Port? - Definition from Techopedia. [Online; accessed 22. Dec. 2022].
- Wahab, F., Zhao, Y., Javeed, D., Al-Adhaileh, M. H., Al-maayyah, S. A., Khan, W., Saeed, M. S., and Kumar Shah, R. (2022). An ai-driven hybrid framework for intrusion detection in iot-enabled e-health. *Computational Intelligence and Neuroscience*, 2022.