Artificial Intelligence Based Malicious Traffic Detection

Lakshmi N. K. Meda and Hamid Jahankhani

Abstract Cyberattacks have become a nightmare for businesses, often having to spend time and resources identifying one or mitigating another. The current research is an effort to develop an artificial intelligence-based security solution that can meet the SME demands of providing a security solution capable of detecting cyberattacks in real-time before they eventually become a crisis for the business. The proposed solution uses multiple layers of deep neural networks using ReLu activation function and Adam algorithm as an optimizer to provide the detection capabilities. While Wireshark provides the model with powerful network monitoring capabilities, Weka fulfils the data pre-processing role to provide the AI module with a clean and structured dataset. The solution is tested for its ability to study the network patterns and capability to distinguish between the regular and malicious traffic. The proposed model is tested using two different datasets, a dataset created in a virtual lab environment, and an IoT-23 dataset. The performance of the proposed AI model is tested on the metrics of 'accuracy' and 'loss'. The model performed well in distinguishing the network traffic on both the datasets. The model will provide the required augmentation capabilities for SMEs to better handle cyber threats.

Keywords Deep neural networks · Rectified linear unit · ReLu · AI · Cyberattack · Adam algorithm · Weka

1 Introduction

Cyberattacks are a buzzword in the world of IT, and to maintain a safe and secure IT infrastructure is often complex, challenging, and requires time, resources, and expert knowledge, all of which are at a premium for Small and Medium Enterprise's (SME). With a natural shortage of resources and dedicated IT personnel, SMEs typically lack

L. N. K. Meda
Northumbria University, London, UK

H. Jahankhani (✉)
Northumbria University London Campus, London, UK
e-mail: Hamid.jahankhani@northumbria.ac.uk

H. Jahankhani et al. (eds.), *Blockchain and Other Emerging Technologies for Digital Business Strategies*, Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-030-98225-6_2

continuous network monitoring leading to a delayed response to security threats. A clear reverse trend is seen in the recent times, as IT Security became a focus area for SMEs. More and more businesses are investing resources in the security domain, but with the current market flooded with variety of security tools, applications, and solutions to deal with various concepts of security, implementing a robust security solution is often unfeasible. To maintain an enterprise grade security, SMEs had to invest in multiple security solutions and still fails to identify an attack.

A network architecture in a business typically involves multiple end user devices surrounding one or more server machines and network devices. The security function is often handled by the individual end-user devices with the network devices often using the pre-configured or default settings. While some businesses use a complete security solution that handles all the devices within the network, most of the networks lack continuous monitoring and had to rely on the alerts produced by the security products. Without a dedicated security personnel monitoring the network activities, these networks infrastructures are prone to malicious attacks. The damage caused by these attacks have more impact on the small and medium businesses than large organisations with 60% of SMEs often close their business within 6 months following an attack [15]. The research aims to design an AI model capable of predicting network attacks by continuously monitoring the network for anomalies. By utilising the deep learning capabilities of AI, the proposed model monitors the network activity and provide an alert when the traffic matches any previously learnt patterns.

2 Literature Review

Artificial Intelligence has the potential to become a revolution in solving problems for a variety of sectors such as healthcare, transport, agriculture, marketing, banking, finance etc. According to Marr [21], the influence of AI in augmenting human capabilities can be seen in all facets of human life. “Artificial Intelligence (AI) is going to change the world more than anything in the history of mankind. More than electricity”, says Dr. Kai-Fu Lee [28].

The importance of artificial intelligence in the field of Cybersecurity has been discussed by Kurpjuhn [19] who emphasises the importance of adding ‘intelligent security capabilities’ to a business’s security infrastructure to counter the ‘proliferating malware and cyberthreats’. The author strongly believes that understanding the vital components that makeup a reliable and robust security solution is key to effectively integrate AI into business IT security rather than mere automation of the security tasks. The author explained about how traditional sandboxing can be boosted by adding AI, to continually analyze and learn from the network traffic, thereby providing robustness to the entire security infrastructure. The author ascertains that importance of up-to-date information that needs to be consistently fed to the security system to maintain ground against zero-day threats which could only be possible by using AI based security solutions. While the author did not provide any practical implementation, he strongly maintained a strong belief that the efficiency

of security solutions in future will largely depend upon the effective integration of artificial intelligence.

The efficiency and need for the use of AI in cybersecurity is discussed by Hofstetter et al. [13]. The authors proposed a hybrid approach of using AI and machine learning to detect malicious activities within a business network. Unlike a fully automated AI solution, the hybrid system adds human input to provide the necessary insights, which according to the authors, not only improves the accuracy and performance but also adds a 'policing aspect' which helps with accountability. A two-phase approach is discussed with (i) phase one dedicated to developing a data-tree with the application of several learning models and human input and (ii) online testing phase two in which the system will provide with a suitable detection plan for a business based on the data-tree. Even though the authors solution is novel, it requires a constant input from human experts to train the system and the accuracy of the system entirely depends on the data-tree which is based on the learning models.

According to Chan et al. [8], the three main benefits of AI in cybersecurity are the detection of false positives, predictive analysis, and the development of an immune system. With the use of Neural Networks and Expert Systems, the authors states that a sophisticated system that can function like human brain while finding solutions from learning models and past data, can be developed. The authors discussed the case studies of Illumio, Blue Talon identify the key requirements of data protection and access and to explain the benefits of using artificial intelligence in such business scenarios. While the AI based cybersecurity systems currently being used are not entirely failsafe with potential ethical concerns, lack of expertise in unsupervised learning etc., the authors state that use of AI in cybersecurity will attain a state where they will be widely accepted and will be more applicable and accessible too.

More thorough research on the capabilities of AI in cybersecurity was put forward by Zeadally et al. [36]. The authors attributed the dire need for improving the current cybersecurity solutions owing to the lack of cyber governance skills, harness the potential of new technologies and fragmented cybersecurity frameworks. The authors discussed the traditional cyberthreats and the legacy security solutions being used to mitigate those threats. The concept of machine learning techniques such as Naïve Bayes, Decision trees, K-nearest neighbours (k-NN), Support Vector Machines (SVMs), Artificial Neural Networks (ANNs), Self-Organising Maps (SOMs) etc. were discussed in-depth about their suitability for use in cybersecurity. The appropriateness and application of the above techniques at various domains such as Internet, Botnets, IOTs, critical infrastructure etc. were critically discussed to explain how the potential of AI in cybersecurity. The authors believe that with further advances in the technology, AI can provide innovative solutions to detect and mitigate sophisticated cyberthreats. The authors statement that AI based machines thinking humanly in future is not overstated.

A framework capable of automatically analysing the patterns of a cyberattack to identify a potential threat to critical infrastructure is proposed as far back as in 2008 by Flammini et al. [11]. The DETECT (Decision Triggering Event Composer and Tracker) framework, developed by the authors allows for triggering a focussed and fully automated response when a threat is detected. By training the system using the

experts knowledge base, improved probability of detection (POD) and situational awareness can be achieved while reducing false positive rates. The authors used Java programming to develop Event Trees in Scenario GUI to detect a threat and trigger a predefined automated response and proved the efficacy of the proposed system using a subway terrorist attack example. While the system appears to be efficient, the practicality in terms of critical infrastructure solely depends on the amount of data that can be analysed at a given point of time and the depth of the security knowledge base that is fed into the framework.

The impact of using Automated Decision Systems (ADS) in cybersecurity was analysed in-depth by Chamberlain et al. [7]. They analysed the ADS systems based on the decision, autonomy, and impact levels. The authors provided comparisons between autonomous systems, AI, and other new technologies such as data science, and states that while AI based systems works well with specialized tasks, they do need human insights and skillset to identify the broader impact and risk. According to the authors, the current level of AI decision systems works primarily based on the analytical parameters, while humans can provide the necessary intuition while making critical decisions. The authors used Stacey's complexity model to identify the relationship between AI and autonomous systems, decisions by humans and machines and considers that AI can augment humans but not completely replace them in the decision making.

A contrasting research of how Artificial Intelligence and Machine Learning can be used both offensively and defensively is carried out by Kamoun et al. [16]. The authors briefly discussed the use of AI/MLS as a defensive mechanism for intrusion detection, network traffic identification, threat detection, digital forensics, botnet and spam detection etc., while the main focus was placed on the offensive uses of AI. By using a System-Fault-Risk (SFR) framework, the authors categorized the cyberattacks as Probe, Scan, Spoof, Flood, Misdirect, Execute and Bypass, and discussed how AI/MLS systems can be used to carry out offensive activities. The authors expressed recommendation that the misuse of AI/MLS models should be proactively anticipated and be considered while developing future solutions. The research discussed the potential of AI systems being used in distinctive sides but lack any practical implementations or recommendations.

Sapavath et al. [30] developed an AI based model to detect cyberthreats and conducted practical tests and evaluations in a virtualized wireless network. The authors used Bayesian network model to develop the test network and used three different data sets to train the model to ensure accuracy of the threat detection. The transmitting power of a user device is taken into consideration to differentiate a malicious user from legitimate users. Whenever the transmitted power is higher than the predefined threshold, the system will generate an alarm and will trigger automated actions to investigate the activity from the specific user. The parameters of Accuracy Score, Recall, Precision, MCC, False-positive rate and G-mean were used to analyze the results. The authors conducted multiple tests and achieved an accuracy of 99.8% in detecting a cyberattack and the proposed model. They compared the results with Deep Neural Network (DNN) and Random Forest models and concluded that their proposed model performed better and used less learning time and achieved better

accuracy levels. The proposed model presents a great start but the performance in a real network needs to be evaluated along with other input parameters alongside the transmitting power.

Thaseen et al. [33] proposed a network traffic classification mechanism without performing decryption of data packets. The proposed approach captures network traffic using Wireshark tool to create a dataset and performs data pre-processing using Weka tool to extract the classification factors necessary for the application of machine learning algorithms. By implementing multiple machine learning algorithms such as NB (naïve bayes), RF (random forest), KNN (KNearest Neighbors) and SVM (support vector machine) the model is tested for predicting and analysis of malicious traffic. The results proved that RF algorithm has the best accuracy in identifying a packet as one of a normal, malicious, normal encrypted or malicious encrypted packet. The authors future work consists of using deep learning models to improve the general performance and classification of network traffic.

One of the comprehensive reviews of using Artificial Intelligence (Deep Learning Methods) in cybersecurity is carried out by Macas and Wu [20]. The authors stated that Deep Learning (DL) models can be highly effective in solving complex cybersecurity problems and set out to review various Deep Learning models while proposing a DL framework of their own. The authors discussed how DL techniques such as RNNs, CNNs, DBNs, DBM, AEs, and GANs can aid in intrusion detection, malware analysis, IoT defence systems etc. The authors proposed a framework with a series of steps involving data collection, pre-processing, feature extraction, evaluation, training and validation, and model selection. The authors state that the input data quality and feature selection play a key role in developing predictors which is vital for machine learning. By dividing the data set into training, validation and a test set, multiple DL models can be tested with the data set with multiple algorithms and specific parameters. The validation set accuracy will help in identifying the ideal model for the network. The chosen model will then be trained using all the available data with the best possible parameters, and will be periodically evaluated to ensure the system can predict zero day attacks. The authors did a commendable work in evaluating various DL models but left several key aspects of data collection, feature extraction, model suitability etc. to the experts who plan to use the framework.

Another key research in identifying the information required to explain the decisions made by AI systems is conducted by Jaigirdar et al. [14]. The authors stressed the importance of identifying the factors that led the AI system to reach a decision, to ensure the system's transparency and accountability. The authors tried to answer four key questions of (i) What information is required to understand a decision made by the AI system, (ii) possibilities of checking solutions accuracy and the decision-making steps to ensure transparency, (iii) adding security-based evidence to detect 'misrepresentation, deliberate bias, safety-mismatch or backdoor', (iv) possibilities of adding parameters to ensure and justify ethical and policy issues. The authors felt the need for inducing a governance and policy factors into AI based systems to ensure complete transparency of the decision making. The authors used PROV-DM data model as a base and developed a 'Six Ws' framework to identify the attributes required to present explainable AI properties. The authors put the framework into

action by using a sample scenario of bank loan processing to find the justification to the decision made by the AI based decision making system. The authors stressed the need for linking the input parameters to the output decisions and deriving an explanation as to why a particular decision is made. The authors came to a conclusion that it is essential to maintain the 'transparency and explainability' in AI systems along with the addition of security and legal attributes and the proposed framework could act as a steppingstone.

An open-ended review and research on identifying the capabilities of intelligent attack detection techniques using artificial intelligence is conducted by Aljabri et al. [3]. The authors started by identifying several classic network attack classification techniques such as port-based, payload-based, Deep packet inspection (DPI), behavioural techniques, rationale-based, Bag of Flow (BoF) etc., all of which depended primarily on the database of pre-defined attack signatures to detect an attack and their shortcomings due to lack of intelligence. The authors prime the importance of intelligent techniques such as machine learning (ML) and Deep learning (DL) and their statistical capabilities in learning and analysing network traffic, thereby identifying network anomalies at a much faster rate compared to non-intelligent techniques. According to the authors, the ability to learn attack scenarios and patterns from a wide variety of sources to train these intelligent systems makes them promising for the future. The authors analysed the current research in Intelligent systems for network attack mitigation using wide variety of techniques based on logistic regression (LR), random forest (RF), decision trees (DT), ensemble of DT, support vector machine (SVM), naïve Bayes (NB), K-nearest neighbor (KNN), K-means clustering etc. The authors identified the trend of using the up-to-date and advanced Machine Learning and Deep Learning techniques of artificial neural network (ANN), recurrent neural network (RNN), convolutional neural network (CNN) and deep neural network (DNN), in network traffic analysis and threat mitigation. The authors categorised all the current research according to the threat type and the possible futuristic solutions. Another important part of the research by Aljabri et al. was to identify the current research trends based on the various datasets available for researchers. The authors stressed the complexity of identifying the right technique along with the right dataset to develop a highly accurate model and agreed that finding a model that works for all types of threats could be a silver bullet if at all one could develop one.

A flow-data based approach to detect and classify malicious network traffic is proposed by Abuadlla et al. [1]. The ease of capturing data from any network device and the scalability offered by aggregated traffic metrics, biased the authors to use flow-data for their proposed model. In a critical anomaly detection phase, malicious traffic is differentiated from normal network traffic and in stage two, the detection and classification module identifies the attack characteristics and classifies the type of attack. To test the proposed IDS system two different neural network methods are chosen, MLFF (Multilayer Feedforward neural network) and RBFN (Radial Basis Function Network). By using multiple training algorithms (Radial Basis Function net, Levenberg-Marquardt and Resilient Backpropagation) the authors were able to achieve a detection rate of 94.2% in anomaly detection phase and a detection rate of 99.42% in classification phase. The model was able to achieve 100% in detecting

DoS and land attacks, and 99.9% in port-scan attacks which shows better performance compared to similar models using large datasets. Although falling behind the multifunction feedforward neural networks in its classification abilities, RBFN neural networks were more suited to real-time networks owing to its simple architecture and hybrid learning capabilities. The authors plan to continue the research of developing a more accurate model with minimal features and less training time.

Yuan et al. [35] sought the use of advanced deep learning techniques to counter DDoS, one of the most harmful network attacks. Called DeepDefense, the authors developed a recurrent neural network learn patterns from the network traffic and to automatically extract high-level features to achieve powerful representation and trace network attack activities. Using the ISCX2012 dataset, the authors selected multiple numerical, Boolean and text fields and used binary, BoW conversions techniques to extract the required features. By designing a Bidirectional Recurrent Neural network and comparing the selected traffic fields against attack vectors the authors were able to differentiate malicious traffic. The authors tested the DeepDefense approach using different RNN models (LSTM, CNNLSTM, GRU, 3LSTM) and concluded that the DeepDefense approach is highly capable in learning from historic network traffic, effective in detecting DDoS and offers better performance in terms of generalization compared to shallow ML models. The use of CNN, RNN, Long Short-Term Memory Neural Network (LSTM) and Gated Recurrent Unit Neural Network (GRU) proved beneficial while training large datasets and helped reduce the error rates by 39 as much as.69% in some scenarios.

Mohammad et al. [24] conducted several experiments using artificial neural networks to identify phishing attacks. Identifying a phishing attack is extremely complicated given the dynamic nature of the websites and hence requires a model that necessitates a constant improvement in the prediction capabilities. The authors automated the network structuring process and conducted experiments that showed resilience against noisy data and fault tolerance while achieving high accuracy. The authors used 17 different features such as ip address, long URL, server form handler, DNS record, age of domain etc., to represent the neurons from a dataset of 1400 phishing and legitimate websites. The research is based on the principle that Phishing detection is a classification problem, and the selected fields were given the values of either “phishy” or “legitimate”. The authors used a neural network with one hidden layer called multi-layered perceptron, in which the number of neurons can be changed to adapt to the complicated relationship between the input and output variables. Although a bit complex, the authors firmly believe that their model will automate the network restructuring process with fewer user inputs and can be adapted to any future updates.

A focus on mitigating zero-day threats using machine learning is carried out by Beaver et al. [6] who states that the ability of machine learning tools to accommodate complex and huge data sizes facilitates a strong combination of data analysis and human augmentation. In the work, they used AdaBoost (adaptive boost) ensemble learner to reliably differentiate malicious network traffic, in a simulation with network settings that mimics a real-time operational network. The model was tested with four levels of decision-making: (1) top-level wrapper placing a cap on training data’s

false-positive rates, a maximum false-positive rate was imposed to 0.01; (2) internal level consisting of AdaBoost ensemble with 1000 rounds of boosting; (3) internal level implementing a decision tree, and (4) an anomaly detection algorithm to identify whether the traffic is normal. Over the 18 experimental runs, the system was able to achieve 94% malicious traffic detection rate and 1.8% false-positive rates. The model was able to detect 89% of the attacks on which the system was not trained which proves the ability to detect zero-day threats. In future, the authors plan to introduce more parallelism to the entire architecture and thereby be able to reach real-time machine learning NIDS capability at 1Gbps.

In a similar research focussing on identifying zero-day botnet attacks in real-time, Ahmed et al. [2] evaluated the accuracy of DNNs (Deep neural network) and determined the capabilities of DNNs by applying the algorithm to a CTU-13 dataset. The proposal was divided into two parts, the first part using a feed-forward back propagation ANN (artificial neural network) and the second part using a DNN, to compare the botnet detection capabilities between deep learning models and traditional machine learning models. By running several experiments with input consisting of multiple feature set, an accuracy of over 99.6% were achieved by using deep learning ANN model with a total loss of 0.54. The accuracy of the deep learning model in detecting a botnet attack is higher than other machine learning models using SVM, Decision Tree and NB. The authors would like to examine the efficacy of the model on alternate datasets and plans to apply deep learning model to detecting other threat types such as DDoS in a future study.

Chou et al. [9] used deep learning algorithms using opensource software tools to develop a system capable of classifying malicious traffic from normal traffic. The research used a deep neural network (DNN) forward propagation algorithm on an NSL-KDD dataset to classify DoS and probing attacks. The model was able to achieve an accuracy of 98.99% in detecting DoS attacks and 97.65% in detecting a probing attack. While the model's performance was as expected on some attacks, it performed poorly in accurately classifying attacks such as U2R (User to Root) and R2L (Remote to Local), which the authors believe is down to monotonous nature of the training dataset causing over-learning. For future, the authors aim to improve upon the training characteristics of the model to overcome the current limitations.

Dutta et al. [10] discussed the issues in anomaly detection in Intrusion Detection Systems using traditional ML models and proposes a stacked generalization approach to achieve reliable classification of outliers. The proposed method utilizes deep models such as DNN, LSTM (Long Short-Term Memory) and meta-classifier (logistic regression) to improve the anomaly detection. The proposed model involves two stages: (1) a data pre-processing stage utilizing a Deep Sparse AutoEncoder (DSAE) which uses a sigmoid instead of neural activation functions; (2) classifier modelling using stacked ensemble (DNN, LSTM) to eliminate bias towards a particular dataset. An assessment of the proposed model is against multiple heterogeneous datasets (IoT-23, LITNET-2020 and NetML-2020) resulted in improved performance compared to other techniques such as RF (Random Forest) and SVM (Support Vector Machine). When validated using pre-specified datasets, the authors found significant improvement in evaluation metrics and can provide the required

accuracy in detecting anomalous behaviour in a network. The authors plan to carry out experiments on sophisticated datasets use advanced computational methods such as Apache Spark to enhance the scalability to cater large network traffic data.

A novel deep learning self-taught learning (STL) based intrusion detection system is proposed by Al-Qatf et al. [4] which helps with dimensionality reduction and feature learning. The model utilizes sparse autoencoder algorithm to improve unsupervised feature learning which reduces the training and testing times and boosts the prediction accuracy. The model is tested using NSL-KDD dataset, the non-numerical features of the dataset are encoded using 1-n system to suit the proposed model before being normalized to map all features. The STL based model proved to be extremely accurate in classifying malicious traffic with significant improvement is training and testing times. In a direct comparison against shallow classification techniques such as J48, naïve Bayesian, RF and SVM, the proposed model showed higher accuracy rate particularly under two-category (normal and attack) and five-category (normal and five attacks) classification. In future research the authors plan to use multiple stages of STl with a hybrid model for better feature representation.

To counter the high false-positive rates which are often accompanied with high accuracy rates in traditional models, considering the spatial and temporal features in the data, Wu and Guo [34] proposed a hierarchical neural network LuNet. LuNet consisted of several layers of convolutional neural networks (CNN) and recurrent neural networks (RNN) which learn in sync from the training data. The CNN + RNN synergy along with increased learning granularity can be utilized to effectively extract spatial and temporal features. CNN often aims at spatial features while RNN targets temporal features and over multiple levels feature extraction becomes spatial oriented. The authors overcame the challenge by introducing a LuNet block which combines both the CNN and RNN blocks at each level thereby retaining all the necessary features. The model is tested on two different non-redundant datasets NSL-KDD and UNSW-NB15 and the design maintained significantly lower false-positives while offering high validation accuracies and detection rates. LuNet makes use of cross-validation scheme to tackle the imbalanced distribution of NSL-KDD dataset. The performance of LuNet is categorized as: (1) Binary Classification in which LuNet distinguishes normal traffic with attack traffic; (2) Multi-Class Classification in which LuNet classifies the traffic as normal or belonging to one type of attack provided in the dataset. In Binary Classification, detection rates of 99.42% and 98.18%, accuracy rates of 99.24% and 97.4% and false-positive rates of 0.53% and 3.96% were achieved for NSL-KDD and UNSW-NB15 datasets. In Multi-Class Classification, the accuracy rates of LuNet averages at 99.05% and 84.98%, detection rates at 98.84% and 95.96% and false-positive rates of 0.65% and 1.89% for the two different datasets. The inefficiency of LuNet in attack classification to some attacks such as backdoors and worms is primarily down to the lack of sufficient samples in the training data and will become a part of authors future work.

The competency of deep learning techniques in detecting cyberattacks was brought to mobile cloud environment by Nguyen et al. [26]. A novel framework was proposed leveraging deep learning algorithms to train a neural network which can detect cyberattacks with high accuracy. All the user requests in the network are

sent through an attack detection module which classifies the traffic and forwards any suspicious packets to the security control module which then verifies the suspicious packet to take appropriate action of either allowing the packet or to block it. By using feature analysis and dimension reduction in the deep learning model, the required features to train the model are extracted. The model is tested on three different datasets NSL-KDD, UNSW-NB15 and KDDcup 1999 to evaluate the performance of the model on the metrics of accuracy, precision and recall, and compare them to other machine learning algorithms such as K-means, RF (random forest classifier), Gaussian Naïve Bayes, Multilayer Perceptron (MLP) etc. The proposed model was able to achieve best performance metrics compared to all other models with an accuracy rate of 90.99%, 95.84 and 97.11% for three datasets. The other evaluation parameters of precision and recall also achieved optimal metrics which demonstrates the stability, robustness, and flexibility of the proposed model. The authors aim to take the model real-time, testing the accuracy on real devices to evaluate the detection times and the power consumption rates.

Mohammad and Alsmadi [23] emphasized the importance of feature selection as a critical component to any AI classification model regardless of the internal algorithm. The efficiency of the selected feature set determines the performance of the classification model; hence the authors proposed an innovative algorithm for feature selection called HW (the Highest Wins). HW uses a statistical approach of measuring the distance between the observed and expected probability values, which is similar to other feature selection algorithms such as X2 (chi-square) and IG (information gain), but is more robust, simple and easy to comprehend. To test the generalization ability of the new method, ten datasets with varying input features were used, and the results showed significant improvement in reducing the dimensionality over other classification models. The evaluation metrics of recall, accuracy, precision and F1 score showed better results using features selected by HW technique, while class imbalance was still an issue similar to X2 and IG. In a second experiment, two versions of NSL-KDD datasets, binary and multiclass, were used. The experiment resulted in performance boost not only in all the evaluation metrics but also in the classification time and number of rules produced. The authors left the class imbalance issue for future work along with identifying advanced search techniques to improve the process of feature selection.

3 Research Methodology

The aim for the research is to identify the feasibility of using artificial intelligence based intrusion detection system in a real-time network. There has been ample amount of research happening in the past few years to develop a fully functional AI model capable of predicting cyberattacks. While most of the research efforts were successful in a way or other, a fully functional model is yet to be designed, or at least is not released publicly as an open source. Most of the researchers focussed on finding the right AI model for implementation across network architectures of

diverse magnitude, which is why the past models are tested using various generalized datasets to identify the threat detection capabilities. There was no research dedicated upon the implementation of AI tailored to the requirements of small and medium scale businesses (SMEs). The current research tries to bridge the gap by introducing an AI component to the traditional security practices used in SMEs as a way to better detect malicious threats. The research tries to follow and utilize some of the methodologies and practices used by other researchers to reach up to speed with the current trends in machine learning implementations. Artificial Intelligence is a broad field of science that comes in various flavours and subsets such as machine learning, natural language processing, expert systems etc. The current research uses a Neural Network (NN) to develop a threat detecting model to cater the needs of an SME.

The research follows a quantitative approach of measuring the performance of the proposed system using multiple evaluation metrics. The research works on the principle that the deep learning model can be integrated to the current security topology in an SME network and is fed with data that flows through the network. The proposed model continuously evaluates the network traffic and triggers an alert whenever the traffic pattern matches a pre-learnt pattern or when the accuracy levels reach a pre-defined threshold.

The research relies on the fact that malicious traffic often exhibits features that are often unique from other traffic and can be identified by carefully choosing the feature set from any given dataset. For example, features such as packet entropy values from source ip address, average arrival time, source bytes can provide the required information to identify a DoS attack [18]. Feature selection plays an important role in a machine learning model in accurately detecting an attack.

3.1 Neural Networks

Neural networks are computing networks consisting of artificial neurons simulating the neurons in human brain that are capable of processing and learning from large amount of data. The arrangement and interconnection between these neurons determine the characteristics of the neural network and its logical problem solving abilities. Neural networks can learn from complex and nonlinear data and can be trained to classify data, identify relationships and patterns, generalize and reason, generate predictions, etc., and the efficiency and accuracy of a neural network relies on the training data that is input to the model.

A typical neural network consists of an input layer, one or more hidden layers, and an output layer. These layers consist of a large number of nodes which are interconnected and have a threshold and weight value. An activation function is used to change the output value beyond the threshold which activates the node and data is passed on to the next level. Abstraction is key as the input data is passed on to several hidden layers where the data processing is performed based on the weight function before passing the data to the output layer. A learning function defines the weight

value which can be increased or decreased to achieve the desired output. Various algorithms exist to understand and correlate the communication between these layers such as Feedforward neural network, convolutional neural network (CNN), Recurrent neural network (RNN) etc.

A simple feedforward neural network technique is used for the current research consisting of multiple hidden layers which are adjusted according to the quantity of the input data.

- Model Topology

The research makes use of multiple tools and technologies to develop and implement an artificial intelligence model capable of detecting malicious threats in an SME environment. The key phases in the whole model are:

- Network monitoring

As the name infers, the network monitoring phase involves data monitoring and capturing. Network monitoring is a critical component in a typical IT network to monitor the network performance and fault tolerance. The research assumes that the network has sensors or monitoring points located at key locations such as servers, firewalls, routers etc. within the network to capture traffic in real-time. The captured data is then fed to the proposed AI model for training and analysis.

- Data pre-processing

A critical aspect of the proposed AI model is data pre-processing which is the process of converting raw data to clean data and is often the first step in working with data. According to Press [29], data scientists often spend around 80% of their time collecting and organising the data. The data collected from the network often unstructured and consists of huge amount of information such as files, audio, video, scripts etc., and machine learning models do not have the required capabilities to understand and process the data. Data pre-processing is critical as incorrect formatting or cleaning often causes more harm than good, and well-organised data is more valuable than the powerful algorithms of machine learning [22].

There are three key aspects of data pre-processing which are:

1. Data cleaning

Raw data can be incomplete, unorganised, and hugely complex for the machine learning models to process. The amount of missing, incomplete and noisy data often adds up to the processing time and accuracy of the proposed model. The data is cleaned of unwanted text, symbols, duplicates, missing data, blanks, etc. in the data cleaning stages.

2. Data transformation

The traffic data captured from the network consists of several fields such as ip addresses, data and time stamps, protocols, packets sizes, number of packets, etc.

There will be large columns of data that are often divided as numerical or categorical. The performance and accuracy of a machine learning model depends entirely on the feature selection, features that are important for the machine learning model. The selected features often dictate the algorithm accuracy, and the data processing time.

The research uses multiple features such as source address, destination address, protocol, source and destination ports, label, label detail and threat as its dataset features.

3. Data reduction

Typical data processing consisting of lot of features costs huge amount of computing resources and time. Data reduction reduces the amount of amount of data in a dataset while maintaining the integrity of the data, i.e., keeping a minimum set of features required for data analysis. Data reduction can be achieved by feature selection and extraction, removing non-essential features, deriving new features by combining existing ones etc. Dimensionality reduction using PCA (Principal component analysis) is one of a key technique where new features are derived from large set of variables.

The current research uses the following principal features: source address, source port, protocol, label, and label_detail.

3.2 Building and Training of the Model

The final part of the proposed model is the area of actual building, training, and testing of the AI model. This part splits the pre-processed data into three types of data: training, validating, and testing data. The model identifies patterns, generates insights, and learn from the training data and validates its knowledge using the validation data. The entire skillset is then tested against the testing data to predict the accuracy level of the model.

In the current research there are two different datasets used for evaluation, a network traffic dataset created in a virtual environment consisting of a server and two other machines, one of them acting as an attacker, and the second dataset is an IoT-23 dataset [12], used to validate the performance and efficiency of the proposed model.

3.3 Tools and Virtual Environments

The research makes use of various open-source tools to design and implement an AI model capable of predicting malicious attacks. The virtual environment is designed using virtual machine program Virtualbox with multiple virtual machines using different flavours of Linux. The test network is as shown in the figure below. The

network consists of a server that is configured to act as a DNS and DHCP roles. The attacker machine is used to carry out malicious tasks while network activity is captured using Wireshark software. For the research simple scan attacks using nmap are performed (Fig. 1).

Some of the other key tools are detailed below.

- TensorFlow/TensorBoard

TensorFlow is an open-source ecosystem of tools, libraries and resources for building and deploying machine learning applications. With a comprehensive set of low-level and high-level APIs (Application Programming Interface), capability of running multiple CPUs or GPUs or mobile platforms, TensorFlow is highly scalable and is an end-to-end platform for developing deep neural network models. TensorFlow is typically programmed using Python, but can be used with a variety of programming languages such as Java, C, C++, R, Matlab etc. TensorFlow provides several pre-trained models and datasets for multiple platforms and these models are suitable for production use which makes it very appealing and well used in various sectors such as healthcare, automobile, image, and face recognition systems, virtual assistant etc. [32].

TensorBoard provides additional capabilities to TensorFlow by providing the visualisations, graphs and measurements needed during a machine learning application workflow. The evaluation metrics used in TensorFlow such as accuracy, loss etc. can be tracked efficiently using TensorBoard.

- Wireshark

Wireshark is a network monitoring and analysis tool that provides customizable in-depth monitoring, auditing, and performance capabilities to organisations. Although being an open source allowing unrestricted use, Wireshark is adaptable, highly stateful, and provides extensive high-fidelity logs for network measurement and analysis. Unlike other monitoring tools, Wireshark offers powerful filters to identify the network issues with ability to use in either graphical mode or TTY mode using

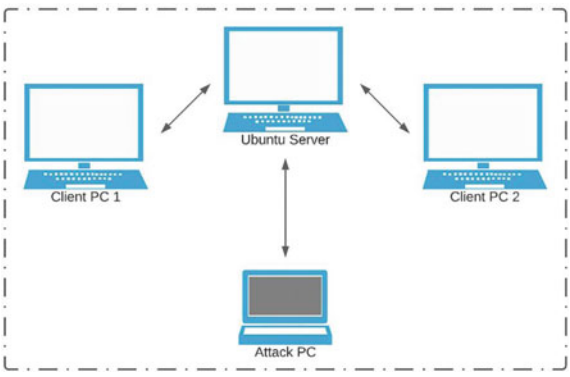

Fig. 1 Virtual lab setup

Tshark utility. Wireshark also provides support with decryption for many protocols such as WPA/WPA2, WEP, IPsec, SSL/TLS etc. The network can be monitored irrespective of the end-devices using wired (Ethernet), wireless (802.11), Bluetooth, frame relay etc.

- IoT-23 Dataset

With a goal of offering large and real dataset for researchers to develop machine learning applications, IoT-23 dataset was captured at Stratosphere Laboratory in Czech Republic. With support from cybersecurity software firm Avast, network traffic from three real IoT devices (Smart LED lamp, Intelligent personal assistant, and a Smart door lock) was captured over a period of two years. The dataset was divided into 23 captures with twenty captures of malicious network traffic and three captures of genuine network traffic. By executing specific malware samples in a controlled environment, the malicious activities were captured, categorised, and labelled accordingly. IoT-23 is a comprehensive dataset with 20 scenarios of data offering more than 280 million flows of malicious data traffic belonging to multiple attack categories such as horizontal port scans, DDoS, Okiru malware etc. With three benign traffic captures providing authentic traffic data to test, IoT-23 presents researchers with excellent resources to conduct machine learning experiments.

3.4 Ethical/Social/Legal Issues

Artificial intelligence is one of the most promising technologies deeply embed with human life and is widely used in various fields such as healthcare, finance, banking, automobiles etc. While AI definitely provide answers to several critical questions of this generation, the ethical, social and legal issues surrounding the use of AI needs more transparency.

- Ethical Issues

The role of ethics in AI is discussed by Anderson and Anderson [5] that AI machines should follow an ethical principle set while making decisions about the actions and procedures to follow, and such as system would find more acceptance than the one without. Ethical issues with AI based systems are often classified depending on whether the systems are considered as subjects or objects [25]. When considering AI as a tool some of the key concerns are data privacy, manipulation, bias, transparency, employment, and autonomy issues, and when AI is considered as a subject has issues with machine ethics and moral agency. Some of other ethical concerns includes misuse, questionable design, and unintended consequences which might determine the effectiveness of AI models in real-time scenarios.

- Social Issues

One of the most delicate and complex questions with artificial intelligence is the application of AI with human like decision-making abilities. Human intelligence often holds reasoning and responsibility at its core and the decisions often include social implications in mind. The social functions applicable to humans such as responsibility, accountability, predictability, incorruptibility transparency etc. needs to be integrated into the already complex AI algorithms and even then, proper governance is required to oversee the implementation. An AI system should be able to gain public trust on its performance and handling which largely depends on maintaining a clear view of responsibility/liability and transparency regarding the accountability. According to Ouchchy et al. [27], providing accurate information access to the public through value statements, factsheets often improve the public trust factor for an enhanced AI adoption.

- Legal Issues

The adaption of AI into the society always begs the question of rationale behind the decision-making process followed by a particular algorithm. The implementation of AI also brings various legal issues surrounding the data protection and privacy, which would have huge implications on the society. A lack of transparency to inspection is a key problem in AI implementations and is often questioned in the face of law.

While the proposed system doesn't have any social, ethical and legal issues might still be applicable with respect to the huge amount of data collected and fed into the model. The information collected from the network will be transformed into binary values to avoid any potential misuse and incorrect representation. Biased feature set selection might become an issue if the design is not governed properly.

3.5 Limitations

The research to develop AI based threat model to detect network attacks is performed completely on virtual environment and, while the performance is as expected and in line with the requirements, the performance when integrated with real networks needs more experimentation and might require considerable changes and testing before it can realize the true capabilities. The use of IoT-23 dataset provides the evaluation metrics to measure the performance even when the AI model is not being used in real-time.

4 Data Analysis and Critical Discussion

The project aims to build an AI tool suitable for SMEs that can detect anomalies in the network traffic. The proposed system can be integrated into a real network system

with data input fed from multiple locations in the network. The tool continuously analyses the network traffic and by using past knowledge learnt from training and testing, predicts whether the traffic is malicious and generates alerts for the security experts. The proposed AI system can be manually monitored or configured to operate automatically according to the recommendations of the AI system.

The performance of the system can be characterized on two important criteria: (a) efficiency of data pre-processing and (b) accuracy and loss metrics of the AI module. There are multiple parameters in each phase of the model that will provide authenticity that the performance of the proposed model is in line with the targeted requirements of the given network. It is equally important to analyse and understand the logical process of the AI model to better gauge the efficiency of the proposed model. The factors that the system is based on are detailed in the following sections.

4.1 Data Pre-processing

The proposed model uses TensorFlow machine learning tool to analyze the data from a csv file, uses its analytical capabilities to measure and compare the new data against the data that was used to train the AI system. Data pre-processing is a key process for the AI model to be effective as the model works entirely based on the data that was fed into the system. The type and quality of data, number of features, size of the dataset etc., forms the core of the data processing stage which in turn affects the AI model's resourcefulness.

- Data Capturing

In the data pre-processing stage, the network traffic is captured using Wireshark network scanning tool. Wireshark allows for traffic analysis at the connection level and can be configured to monitor for activity based on the protocol such as ICMP (Internet Control Message Protocol), TCP (Transmission Control Protocol), UDP (User Datagram Protocol) etc. The network designed in the virtual environment has a server and multiple client machines and the Wireshark tool is run on the attack machine.

The AI tool TensorFlow reads data from files with extension of CSV (Comma Separated Values) and hence the data captured by the Wireshark tool is saved as a CSV file enabling the AI model to extract the data contained in the file.

Data Formatting and Organizing

The pcap files that are captured holds various fields of data such as source and destination addresses, protocols, time stamps, etc. The details of a pcap file converted into csv are as shown in Fig. 2.

The same information for a data file from an IoT-23 dataset is as shown in Fig. 3. The IoT-23 dataset shown is already cleaned, organised, and labelled. As seen in the figure below, the dataset contains many featured columns which can provide

22	23.01194323	fe80::480d:8efc:4951:3ec4	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1	
23	23.98793857	192.168.56.26	192.168.56.22	ICMP	98	Echo (ping) request id=0x6d79, seq=2/512, ttl=64 (reply in 24)	
24	23.98882465	192.168.56.22	192.168.56.26	ICMP	98	Echo (ping) reply id=0x6d79, seq=2/512, ttl=64 (request in 23)	
25	23.98909203	192.168.56.26	192.168.56.22	DNS	86	Standard query 0x8995 PTR 22.56.168.192.in-addr.arpa	
26	23.99017251	192.168.56.22	192.168.56.26	DNS	133	Standard query response 0x8995 PTR 22.56.168.192.in-addr.arpa PTR mail.testlab.co PTR as1.testlab.co	
27	24.98967964	192.168.56.26	192.168.56.22	ICMP	98	Echo (ping) request id=0x6d79, seq=3/768, ttl=64 (reply in 28)	
28	24.99028872	192.168.56.22	192.168.56.26	ICMP	98	Echo (ping) reply id=0x6d79, seq=3/768, ttl=64 (request in 27)	
29	24.99057869	192.168.56.26	192.168.56.22	DNS	86	Standard query 0x2ba7 PTR 22.56.168.192.in-addr.arpa	
30	24.99134663	192.168.56.22	192.168.56.26	DNS	133	Standard query response 0x2ba7 PTR 22.56.168.192.in-addr.arpa PTR as1.testlab.co PTR mail.testlab.co	
31	25.99228491	192.168.56.26	192.168.56.22	ICMP	98	Echo (ping) request id=0x6d79, seq=4/1024, ttl=64 (reply in 32)	
32	25.99784925	192.168.56.22	192.168.56.26	ICMP	98	Echo (ping) reply id=0x6d79, seq=4/1024, ttl=64 (request in 31)	
33	25.99809629	192.168.56.26	192.168.56.22	DNS	86	Standard query 0x7651 PTR 22.56.168.192.in-addr.arpa	
34	25.9998853	192.168.56.22	192.168.56.26	DNS	133	Standard query response 0x7651 PTR 22.56.168.192.in-addr.arpa PTR as1.testlab.co PTR mail.testlab.co	
35	27.01488557	fe80::480d:8efc:4951:3ec4	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1	
36	28.01348909	PcsCompu_1f:30:76	PcsCompu_44:a6:86	ARP	42	Who has 192.168.56.22? Tell 192.168.56.26	
37	28.01416588	PcsCompu_44:a6:86	PcsCompu_1f:30:76	ARP	60	192.168.56.22 is at 08:00:27:44:a6:86	
38	28.05169825	PcsCompu_44:a6:86	PcsCompu_1f:30:76	ARP	60	Who has 192.168.56.26? Tell 192.168.56.22	
39	28.05172729	PcsCompu_1f:30:76	PcsCompu_44:a6:86	ARP	42	192.168.56.26 is at 08:00:27:1f:30:76	

Fig. 2 Traffic capture in VM dataset

#fields	ts	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	proto	service	duration	orig_bytes	resp_bytes	conn_state	local_orig	local_resp	missed_bytes	history	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_bytes	tunnel_parents	label	detailed-label
#types	time	string	addr	port	addr	port	enum	string	interval	count	count	string	bool	bool	count	string	count	count	count	count	set[string]	string	string
1.53E+09	CbJLGOox	192.168.1(	123	81.2.254.2	123	udp	-	0.00549	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	CF3cnH4jD	192.168.1(	123	147.231.1(	123	udp	-	0.001741	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	CigmSt3br	192.168.1(	123	31.31.74.3	123	udp	-	0.004495	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	Cav32mAc	192.168.1(	123	147.251.4(	123	udp	-	0.000988	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	CIvPFAAD	192.168.1(	123	147.231.1(	123	udp	-	0.001487	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	CsayIRCZT	192.168.1(	123	37.157.19(	123	udp	-	0.004739	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	CapItr2s2r	192.168.1(	123	147.231.1(	123	udp	-	0.001743	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	CqXbOUQ	192.168.1(	123	31.31.74.3	123	udp	-	0.007737	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	CTCAcUsI	192.168.1(	123	147.231.1(	123	udp	-	0.001493	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	CHtZjyvKa	192.168.1(	123	80.79.25.1	123	udp	-	0.002248	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-
1.53E+09	Cuz5ANgJ	192.168.1(	123	37.157.19(	123	udp	-	0.004997	48	48	SF	-	-	0	Dd	1	76	1	76	(empty)	Benign	-

Fig. 3 Traffic capture in IoT-23 dataset

additional information regarding the network but are not appropriate for the current research.

The csv file after removing unwanted columns and changing the labels for the IoT-23 dataset is shown in Fig. 4. The total number of columns are reduced and the label set is completely redone to provide a clear understanding and ease of use while working further on the dataset.

- **Feature Selection**

The csv file consists of multiple fields of data organised using the tools described above. The csv files are structured, labelled and tab-organized and ready for the AI model to be trained upon. The final part of the data pre-processing is the feature selection which is the key for the AI model to identify malicious activity accurately. The features must be relevant, fit the purpose and is often considered the core of the machine learning [31].

time	source_address	source_port	destination_address	destination_port	protocol	orig_pa	label	label_detail	threat
CfHl9r3XN	192.168.100.103	34243	147.7.65.203	49560	tcp	180	Benign -		0
CwxSC63a	192.168.100.103	43763	154.8.94.65	14336	udp	40	Benign -		0
CUmrqr4s	192.168.100.103	51524	65.127.233.163	23	tcp	180	Malicious	PartOfAHorizontalPortScan	1
CH98aB3s	192.168.100.103	56305	63.150.16.171	23	tcp	60	Malicious	PartOfAHorizontalPortScan	1
C3GBTkIN	192.168.100.103	41101	111.40.23.49	23	tcp	60	Malicious	PartOfAHorizontalPortScan	1
CC6vK11d	192.168.100.103	43763	196.170.198.141	11764	udp	40	Benign -		0
CDe43c1P	192.168.100.103	60905	131.174.215.147	23	tcp	180	Malicious	PartOfAHorizontalPortScan	1
CJaDcG3M	192.168.100.103	44301	91.42.47.63	23	tcp	60	Malicious	PartOfAHorizontalPortScan	1
CMBrup3E	192.168.100.103	50244	120.210.108.200	23	tcp	60	Malicious	PartOfAHorizontalPortScan	1
C7USrA15	192.168.100.103	34840	145.164.35.6	21288	tcp	60	Benign -		0
Cwm6z43	192.168.100.103	43763	118.112.204.139	2441	udp	40	Benign -		0

Fig. 4 Modified traffic capture csv file in IoT-23 dataset

The feature selection process reduces the unnecessary columns in the dataset which have little or no impact on the model's output and are often considered as noise. The feature selection can be automated using feature selection algorithms such as filter methods, wrapper methods, and embedded methods. A good pre-processed dataset improves accuracy, reduces overfitting and training time.

The feature or attribute selection for the current model is carried out using Weka, a tool which has the tools for data preparation for AI models. The csv file generated in the previous section is loaded into the Weka tool which allows for several machine learning algorithms to be implemented on the data. The 'ClassifierAttributeEval' algorithm for attribute classifier with 'Ranker' search method is used to identify the best feature set in the dataset. The process can be seen in Fig. 5.

As seen in the figure above the attributes that makes a difference in the AI model's performance are identified according to their ranking order. As identified above, the label_detail holds the utmost value which identifies whether the given packet belongs to a malicious traffic or genuine network traffic. Weka allows for several algorithms to be tested on the data before concluding on a suitable algorithm for the given network requirement, and for the current model the 'ClassifierAttributeEval' algorithm provides the best result.

```
=== Run information ===

Evaluator:     weka.attributeSelection.ClassifierAttributeEval -execution-slots 1 -B weka.classifiers.rules.ZeroR -F 5 -T 0.01 -R 1 -E DEFAULT --
Search:        weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:      1-1dataset-weka.filters.unsupervised.attribute.Remove-R1,7
Instances:     1008748
Attributes:    8
               source_address
               source_port
               destination_address
               destination_port
               protocol
               label
               label_detail
               threat
Evaluation mode:    evaluate on all training data


=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (numeric): 8 threat):
        Classifier feature evaluator

        Using  Wrapper Subset Evaluator
        Learning scheme: weka.classifiers.rules.ZeroR
        Scheme options:
        Subset evaluation: RMSE
        Number of folds for accuracy estimation: 5


Ranked attributes:
  0   7 label_detail
  0   3 destination_address
  0   2 source_port
  0   4 destination_port
  0   6 label
  0   5 protocol
  0   1 source_address

Selected attributes: 7,3,2,4,6,5,1 : 7
```

Fig. 5 ClassifierAttributeEval algorithm using Weka

4.2 *Artificial Intelligence System*

During the data pre-processing phase, the network traffic data is collected, organized structurally into tab-separated CSV (Comma Separated Values) files. Depending on the amount of network traffic, the data is fed into the TensorFlow as a single csv file or multiple csv files. The system reads the data, carries out the functions defined in the TensorFlow code, and calculates the accuracy which is the measure against the pre-learnt knowledge from the training data.

- TensorFlow Code

TensorFlow allows for the creation of AI models with several layers of neural networks and is capable of processing large amounts of data within these neural network layers. The entire process is extremely resource intensive and requires the use of dedicated computing infrastructure capable of running extremely complex parallel processes. The current model is hence compiled using Google Colab, a cloud environment that gives developers access to high-end computing resources.

The code used for TensorFlow AI model is programmed using Python language and is as shown in the picture in Fig. 6. The initial part of the code deals with importing the required python libraries that are needed for running the TensorFlow model. The next part deals with mounting a cloud drive which stores all the csv files generated during the pre-processing stage. The model is capable and coded to learn from a single csv file or from multiple csv data files. The features set from the csv files are then organised and categorised according to the selected feature set.

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import feature_column
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from pandas.api.types import CategoricalDtype
#Use Pandas to create a dataframe
#In windows to get file from path other than same run directory see:
#https://stackoverflow.com/questions/16952632/read-a-csv-into-pandas-from-f-drive-on-windows-7
dataframe = pd.read_csv("/home/lakshmi/Desktop/CSV/combineddataset2.csv")
#print(dataframe.head())
#show dataframe details for column types
#print(dataframe.info())
#print(pd.unique(dataframe['user']))
#https://pbpython.com/categorical-encoding.html
dataframe["source_address"] = dataframe["source_address"].astype('category')
dataframe["source_port"] = dataframe["source_port"].astype('category')
dataframe["label"] = dataframe["label"].astype('category')
dataframe["source_address_cat"] = dataframe["source_address"].cat.codes
dataframe["source_port_cat"] = dataframe["source_port"].cat.codes
dataframe["label_cat"] = dataframe["label"].cat.codes
#print(dataframe.info())
#print(dataframe.head())
#save dataframe with new columns for future datmapping
dataframe.to_csv('dataframe-export-allcolumns.csv')
#remove old columns
del dataframe["source_address"]
del dataframe["source_port"]
del dataframe["label"]
#restore original names of columns
dataframe.rename(columns={"source_address_cat": "source_address", "source_port_cat": "source_port", "label_cat": "label"}, inplac
print(dataframe.head())
print(dataframe.info())
```

Fig. 6 Python code for VM dataset, Part-1

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
File names: ['/content/drive/My Drive/AI/Dataset/1-1dataset.csv', '/content/drive/My Drive/AI/Dataset/3-1dataset.csv', '/content/drive/My
      time  source_address  ...  destination_port  orig_packets
0  2179639            7194  ...             49480            68
1  3114058            7194  ...             14308             1
2  1626229            7194  ...                23            68
3   907023            7194  ...                23             6
4   172311            7194  ...                23             6

[5 rows x 10 columns]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3274347 entries, 0 to 1374
Data columns (total 10 columns):
 #   Column               Dtype
---  ------               -----
 0   time                 int32
 1   source_address       int16
 2   destination_address  int32
 3   protocol             int8
 4   label                int8
 5   label_detail         int8
 6   threat               int8
 7   source_port          int32
 8   destination_port     int32
 9   orig_packets         int16
dtypes: int16(2), int32(4), int8(4)
memory usage: 99.9 MB
None
```

Fig. 7 Python Code output for VM dataset

The output for the TensorFlow code above is shown in Fig. 7. The code shows the feature imported by the program from the csv data files along with the column types and other information in the data file.

The second part of the code is when the artificial intelligence is put to work on the data imported from the csv data files. The code is detailed in the figure below which starts with splitting the large amount of data into multiple categories of training, testing and validation. The key for this model is the 'Threat' column, which identifies the traffic as benign or malicious and learns the characteristics of such threat type by looking at data from other feature columns in the same row. The dataframe works by reading line by line data and studying the type of information contained in the fields for the corresponding benign or malicious traffic type (Fig. 8).

The last part of the code builds, compiles, and trains a sequential AI model using TensorFlow functions. An activation function called ReLU (Rectified Linear Unit) is used to create a network with multiple layers, the details of the layers are shown below in the figure. The model uses a stochastic optimizer algorithm 'Adam' [17] to test the model in 'accuracy' and 'loss' metrics. A stream of continuous data is input to the system and higher value of accuracy determines that the network traffic data is malicious. The model is run for five iterations or epochs to determine the accuracy levels.

Figure 9 shows the neural network layers for the proposed AI model.

```
#Split the dataframe into train, validation, and test
train, test = train_test_split(dataframe, test_size=0.2)
train, val = train_test_split(train, test_size=0.2)
print(len(train), 'train examples')
print(len(val), 'validation examples')
print(len(test), 'test examples')
#Create an input pipeline using tf.data
# A utility method to create a tf.data dataset from a Pandas Dataframe
def df_to_dataset(dataframe, shuffle=True, batch_size=32):
  dataframe = dataframe.copy()
  labels = dataframe.pop('threat')
  ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
  if shuffle:
    ds = ds.shuffle(buffer_size=len(dataframe))
  ds = ds.batch(batch_size)
  return ds
#choose columns needed for calculations (features)
feature_columns = []
for header in ["source_address", "field_type", "label"]:
    feature_columns.append(feature_column.numeric_column(header))
#create feature layer
feature_layer = tf.keras.layers.DenseFeatures(feature_columns)
#set batch size pipeline
batch_size = 32
train_ds = df_to_dataset(train, batch_size=batch_size)
val_ds = df_to_dataset(val, shuffle=False, batch_size=batch_size)
test_ds = df_to_dataset(test, shuffle=False, batch_size=batch_size)
#create compile and train model
model = tf.keras.Sequential([
  feature_layer,
  layers.Dense(128, activation='relu'),
  layers.Dense(128, activation='relu'),
  layers.Dense(1)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.fit(train_ds,
          validation_data=val_ds,
          epochs=5)
loss, accuracy = model.evaluate(test_ds)
print("Accuracy", accuracy)
```

Fig. 8 Python Code for VM Dataset, Part-2

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_features (DenseFeatur  multiple                 0
 es)

 dense (Dense)               multiple                  768

 dense_1 (Dense)             multiple                  16512

 dense_2 (Dense)             multiple                  129

=================================================================
Total params: 17,409
Trainable params: 17,409
Non-trainable params: 0
```

Fig. 9 Neural network layers

4.3 AI Model Analysis

The AI model proposed in this research is tested on a network simulation in a virtual environment and the model's performance is verified using alternate IoT-23 dataset. The results of the AI system are discussed below:

Analysis of AI model using VM dataset

The dataset from the virtual machine setup is tested on the AI system and the output is as shown in Fig. 10.

The dataset contains a total of 1,008,750 flows of data which are captured while performing malicious networks acts from the attack machine. The dataset is split into 645,600 samples of training data, 161,400 examples of validation data and 201,750 flows of data acting as test examples. The AI model uses the training data to generate insights and acquire knowledge which is validated using the validation samples. The system tests the knowledge acquired using the test dataset before calculating the accuracy and loss parameters.

The proposed system reached an accuracy level of 46.74% with an average loss of 0.69% over 5 epochs. The low accuracy level can be attributed to the similar

```
Consider rewriting this model with the Functional API.
20175/20175 [==============================] - 97s 4ms/step - loss: 736507.6250 - accuracy: 0.4996 - val_loss: 0.6905 - val_acc
uracy: 0.4633
Epoch 2/5
20175/20175 [==============================] - 102s 4ms/step - loss: 0.6908 - accuracy: 0.4650 - val_loss: 0.6905 - val_accurac
y: 0.4633
Epoch 3/5
20175/20175 [==============================] - 108s 5ms/step - loss: 0.6908 - accuracy: 0.4650 - val_loss: 0.6905 - val_accurac
y: 0.4633
Epoch 4/5
20175/20175 [==============================] - 108s 5ms/step - loss: 0.6908 - accuracy: 0.4650 - val_loss: 0.6905 - val_accurac
y: 0.4633
Epoch 5/5
20175/20175 [==============================] - 106s 4ms/step - loss: 0.6907 - accuracy: 0.4650 - val_loss: 0.6904 - val_accurac
y: 0.4633
6305/6305 [==============================] - 22s 3ms/step - loss: 0.6911 - accuracy: 0.4674
Accuracy 0.4674398899078369
```

Fig. 10 AI Model performance for VM dataset

numbers of malicious and benign traffic packets. The dataset consists of a smaller number of threat samples compared to regular network traffic which affected the accuracy levels. Better accuracy levels are possible when the model is trained with more malicious samples with diverse training samples and attack versions.

The AI model performed satisfactorily for the given virtual environment, but the accuracy levels are nowhere close to make any sound decisions in terms of the malicious nature of the network traffic, especially in a real-time network. The model is hence tested using another dataset captured using real devices with diverse attacks and over several years, which is IoT-23 dataset. The working of the mode using a diverse and large dataset such as IoT-23 is discussed below.

4.4 Analysis of AI Model Using IoT-23 Dataset

The IoT-23 dataset consists of traffic captures from 3 real IoT hardware devices connected to a network with access to the internet. The data is captured in 20 different scenarios using 20 different malware samples, and the entire dataset is clearly labelled with the malware types used within the respective scenarios. The IoT-23 dataset used in the current research makes use of 8 scenarios of data consisting of millions of data traffic flows.

The IoT-23 dataset is input to the proposed AI model to test the performance of the model and to ascertain its capabilities in predicting malicious traffic. Figures 11 and 12 shows the python code changed to adapt to the multiple csv files from the IoT-23 dataset. Unlike the AI model that is run in a virtual environment for the first part of the performance analysis, the AI model is run on a Google Colab cloud environment due to the processing power required to handle the large number of data flow samples in the IoT-23 dataset.

Figure 13 shows the computational graph that shows the data flow through the layers in the AI model. The graph shows how the optimizer and other metrics are used.

The performance of the AI model on the IoT-23 dataset is shown in Fig. 14.

As seen in the output window, the proposed TensorFlow model achieved excellent results in terms of accuracy and loss. When given a large dataset with enough samples to train and test, the model consistently showed accuracy levels of over 99% with loss values confined to 0.0093%. The model performed exceptionally showing the capabilities of the proposed model when given a large dataset with enough data samples.

Fig. 11 Updated TensorFlow Code for IoT-23 Dataset, Part 1

```
import numpy as np
import pandas as pd
import tensorflow as tf
import glob
from tensorflow import feature_column
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from pandas.api.types import CategoricalDtype
from tensorflow.keras.callbacks import TensorBoard
from google.colab import drive
drive.mount('/content/drive')

#Use Pandas to create a dataframe
path = "/content/drive/My Drive/AI/Dataset/"
files = glob.glob(path + "/*.csv")
print('File names:', files)
dataframe = pd.DataFrame()
content = []

# To check all the csv files in the path
for filename in files:
    df = pd.read_csv(filename, index_col=None)
    content.append(df)

dataframe = pd.concat(content)
```

4.5 Performance Analysis Using TensorBoard

The artificial intelligence model's performance is analysed further to identify the performance concentrations over the processing period. To perform additional analysis, it is important to track and evaluate model, the process flow, hence advanced analytical view of the whole is essential. TensorFlow comes with an add-on web application called TensorBoard which provides the necessary visualisations to keep track of the model's metric such as learning rate, loss etc.

TensorBoard is an external process and hence needs to be called by the TensorFlow machine learning process. Once called, TensorBoard runs simultaneously along with TensorFlow process and logs all the events, LogDirs, execution summaries, process values etc., which are then presented in a variety of ways such as distributions, graphs, histograms etc. The python code for the proposed AI model is modified to accommodate the TensorBoard calling process.

The following figure shows the Scalars in TensorBoard which shows the key metrics of accuracy and loss and their performance over each of the epochs (iterations). As seen in the figure the trend of validation data is upwards which proves that the model is learning well, but at epoch 3 and 9, the model is slightly overfitting which needs to be weight adjusted. Typically, an AI model is run for several hundreds of epochs as the machine learning curve increases by running the model for more iterations which holds true to the current model. The training and validation are expected to reach a steady upward curve by running the algorithm for more epochs.

```
#print(dataframe.head())
#show dataframe details for column types
#print(dataframe.info())
#print(pd.unique(dataframe['user']))
#https://phpython.com/categorical-encoding.html
dataframe["time"] = dataframe["time"].astype('category')
dataframe["source_address"] = dataframe["source_address"].astype('category')
dataframe["destination_address"] = dataframe["destination_address"].astype('category')
dataframe["protocol"] = dataframe["protocol"].astype('category')
dataframe["label"] = dataframe["label"].astype('category')
dataframe["label_detail"] = dataframe["label_detail"].astype('category')
dataframe["threat"] = dataframe["threat"].astype('category')
dataframe["source_port"] = dataframe["source_port"].astype('category')
dataframe["destination_port"] = dataframe["destination_port"].astype('category')
dataframe["orig_packets"] = dataframe["orig_packets"].astype('category')


dataframe["time_cat"] = dataframe["time"].cat.codes
dataframe["source_address_cat"] = dataframe["source_address"].cat.codes
dataframe["destination_address_cat"] = dataframe["destination_address"].cat.codes
dataframe["protocol_cat"] = dataframe["protocol"].cat.codes
dataframe["label_cat"] = dataframe["label"].cat.codes
dataframe["label_detail_cat"] = dataframe["label_detail"].cat.codes
dataframe["threat_cat"] = dataframe["threat"].cat.codes
dataframe["source_port_cat"] = dataframe["source_port"].cat.codes
dataframe["destination_port_cat"] = dataframe["destination_port"].cat.codes
dataframe["orig_packets_cat"] = dataframe["orig_packets"].cat.codes
#dataframe["label_detail"] = dataframe["label_detail"].astype('string')
#print(dataframe.info())
#print(dataframe.head())
#save dataframe with new columns for future datmapping
dataframe.to_csv('dataframe-export-allcolumns.csv')
#remove old columns
del dataframe["time"]
del dataframe["source_address"]
del dataframe["destination_address"]
del dataframe["protocol"]
del dataframe["label"]
del dataframe["label_detail"]
del dataframe["source_port"]
del dataframe["destination_port"]
del dataframe["threat"]
del dataframe["orig_packets"]
#restore original names of columns
dataframe.rename(columns={"source_address_cat": "source_address", "destination_address_cat": "destination_address",
                          "time_cat": "time", "protocol_cat": "protocol", "label_detail_cat": "label_detail", "label_cat":
                          "label", "threat_cat": "threat", "source_port_cat": "source_port", "destination_port_cat":
                          "destination_port", "orig_packets_cat": "orig_packets"}, inplace=True)
print(dataframe.head())
print(dataframe.info())
#save dataframe cleaned up
dataframe.to_csv('dataframe-export-int-cleaned.csv')

#dataframe = np.asarray(dataframe).astype(np.float32)
#dataframe = dataframe.astype(np.float32)
#tf.convert_to_tensor(dataframe, dtype=tf.int32)
#Split the dataframe into train, validation, and test
train, test = train_test_split(dataframe, test_size=0.2)
train, val = train_test_split(train, test_size=0.2)
print(len(train), 'train examples')
print(len(val), 'validation examples')
print(len(test), 'test examples')
#Create an input pipeline using tf.data
# A utility method to create a tf.data dataset from a Pandas Dataframe
def df_to_dataset(dataframe, shuffle=True, batch_size=32):
  dataframe = dataframe.copy()
  labels = dataframe.pop('threat')
  ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
  if shuffle:
    ds = ds.shuffle(buffer_size=len(dataframe))
  ds = ds.batch(batch_size)
  return ds
#choose columns needed for calculations (features)
feature_columns = []
for header in ["source_address", "source_port", "protocol", "label", "label_detail"]:
    feature_columns.append(feature_column.numeric_column(header))
#create feature layer
feature_layer = tf.keras.layers.DenseFeatures(feature_columns)
#set batch size pipeline
batch_size = 32
train_ds = df_to_dataset(train, batch_size=batch_size)
val_ds = df_to_dataset(val, shuffle=False, batch_size=batch_size)
test_ds = df_to_dataset(test, shuffle=False, batch_size=batch_size)

#create tensorboard callback
tensorboard_callback = [TensorBoard(
    log_dir="logs",
    histogram_freq=1,
    write_graph=True,
    write_images=False,
    update_freq="epoch",
)]

#create compile and train model
model = tf.keras.Sequential([
  feature_layer,
  layers.Dense(128, activation='relu'),
  layers.Dense(128, activation='relu'),
  layers.Dense(1)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.fit(train_ds,
          validation_data=val_ds,
          epochs=10, callbacks=[tensorboard_callback])
loss, accuracy = model.evaluate(test_ds)
print("Accuracy", accuracy)
```

Fig. 12 Updated TensorFlow Code for IoT-23 dataset, Part 2

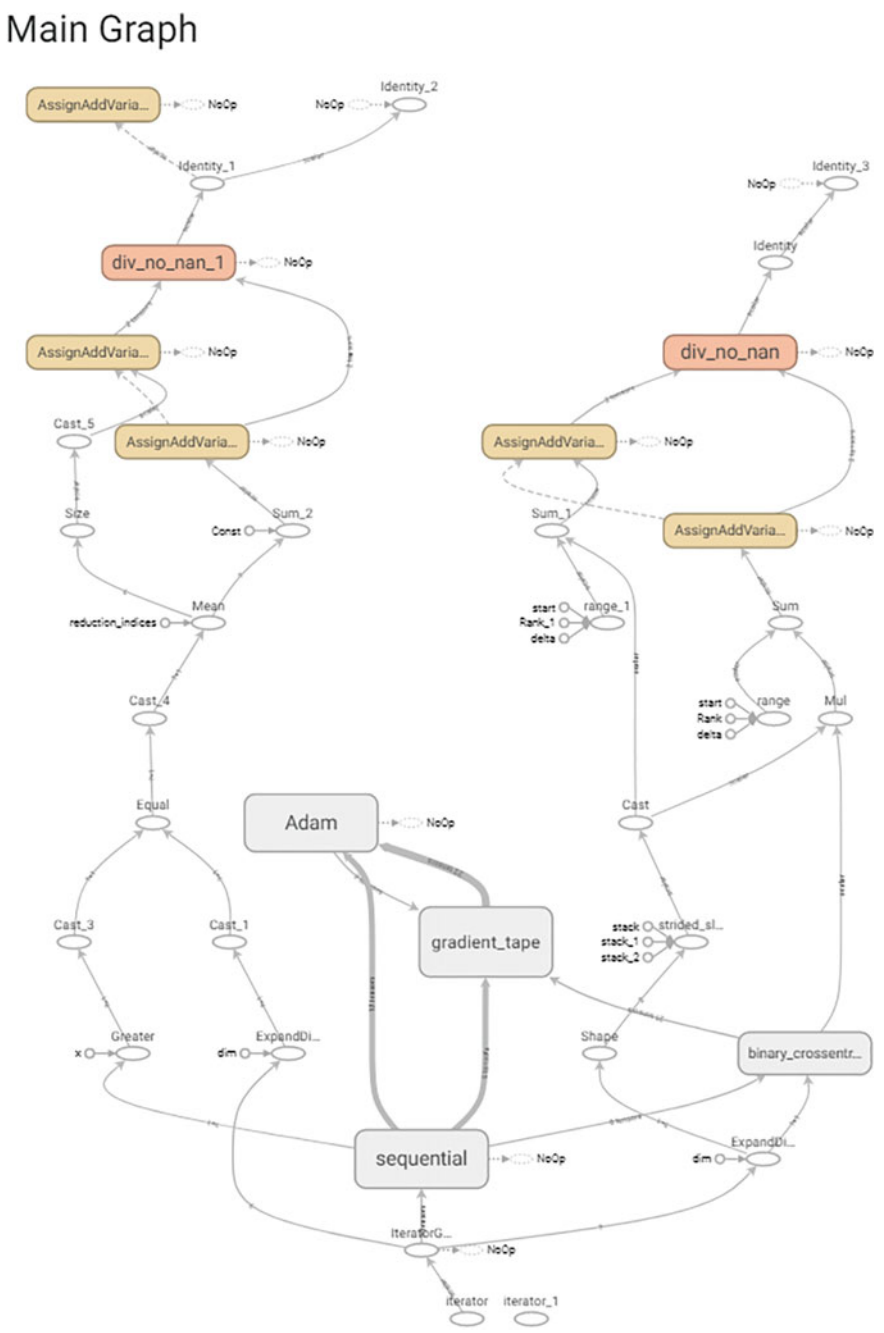

Fig. 13 TensorBoard Graph showing AI code flow

```
2095581 train examples
523896 validation examples
654870 test examples
Epoch 1/10
WARNING:tensorflow:Layers in a Sequential model should only have a single input tensor, but we receive a <class 'dict'> input: {'time': <tf.Tensor 'Iterator(
Consider rewriting this model with the Functional API.
WARNING:tensorflow:Layers in a Sequential model should only have a single input tensor, but we receive a <class 'dict'> input: {'time': <tf.Tensor 'Iterator(
Consider rewriting this model with the Functional API.
65487/65487 [==============================] - ETA: 0s - loss: 1.3995 - accuracy: 0.9748WARNING:tensorflow:Layers in a Sequential model should only have a si
Consider rewriting this model with the Functional API.
65487/65487 [==============================] - 250s 4ms/step - loss: 1.3995 - accuracy: 0.9748 - val_loss: 0.0028 - val_accuracy: 0.9984
Epoch 2/10
65487/65487 [==============================] - 272s 4ms/step - loss: 0.0150 - accuracy: 0.9967 - val_loss: 3.2167e-04 - val_accuracy: 1.0000
Epoch 3/10
65487/65487 [==============================] - 271s 4ms/step - loss: 0.0136 - accuracy: 0.9971 - val_loss: 8.6603e-04 - val_accuracy: 1.0000
Epoch 4/10
65487/65487 [==============================] - 246s 4ms/step - loss: 0.0914 - accuracy: 0.9596 - val_loss: 0.0147 - val_accuracy: 0.9954
Epoch 5/10
65487/65487 [==============================] - 260s 4ms/step - loss: 0.0251 - accuracy: 0.9924 - val_loss: 0.0107 - val_accuracy: 0.9980
Epoch 6/10
65487/65487 [==============================] - 271s 4ms/step - loss: 0.0114 - accuracy: 0.9972 - val_loss: 0.0029 - val_accuracy: 0.9991
Epoch 7/10
65487/65487 [==============================] - 251s 4ms/step - loss: 0.0154 - accuracy: 0.9949 - val_loss: -1.3915e-04 - val_accuracy: 1.0000
Epoch 8/10
65487/65487 [==============================] - 243s 4ms/step - loss: 0.0094 - accuracy: 0.9983 - val_loss: 0.0083 - val_accuracy: 0.9980
Epoch 9/10
65487/65487 [==============================] - 270s 4ms/step - loss: 0.0088 - accuracy: 0.9981 - val_loss: -1.6733e-04 - val_accuracy: 1.0000
Epoch 10/10
65487/65487 [==============================] - 262s 4ms/step - loss: 0.0526 - accuracy: 0.9796 - val_loss: 0.0102 - val_accuracy: 0.9980
20465/20465 [==============================] - 53s 3ms/step - loss: 0.0093 - accuracy: 0.9982
Accuracy 0.998204231262207
Model: "sequential"
```

Fig. 14 TensorFlow Output for IoT-23 dataset

Figure 15 also shows the loss metric over each epoch. As the model is run multiple times the loss value decreased substantially within 4 epochs before settling at the bottom with rates close to 0.006%, which is close to the bare minimum of lose values.

Figures 16 and 17 shows the TensorBoard distributions and histograms i.e., the statistical distributions of the tensors over time. The distributions identify the weight changes in the model over the entire processing period.

Data Analysis

The aim of the current research is to develop a malicious traffic identifying Artificial Intelligence tool that can be integrated to a real network. The proposed AI model produced the results that look promising in achieving the goal of accurately identifying malicious traffic in a network. While the current model is yet to be tested on a real network, the accuracy levels achieved using the current model does look assuring for implementing in a real network.

The key phase of the AI model is the data pre-processing during which the network traffic captured is cleaned, optimized, and structured to befit machine learning model's input requirements. The current dataset was captured with more than 25 columns of data but was reduced to 8 featured columns holding the key attributes of source and destination address, source and destination port, protocol, label and label detail and threat. To verify the attributes and their effectiveness, the Weka tool is run with multiple algorithms such as 'CorrelationAttributeEval', 'GainRatioAttributeEval','WrapperSubsetEval' etc. before finalising on 'ClassifierAttributeEval' algorithm which is better suited to the current AI model design. The attributes are ranked giving the AI model a head start in identifying the key features for the AI model to generate the best possible evaluation metrics.

The AI model achieved low accuracy while implemented on a dataset created using a virtual lab environment. The VM data set contains multiple fields with the necessary features such as source and destination addresses, protocols etc., and contains traffic for port scanning, information gathering attacks. While the dataset is fairly large, the

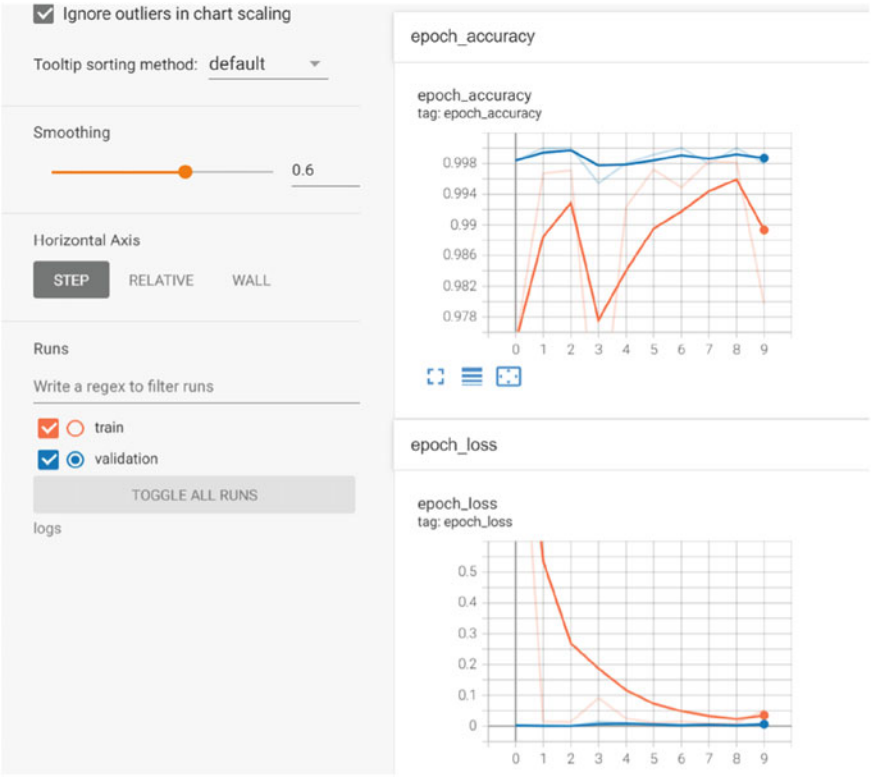

Fig. 15 AI Model performance analysis using TensorBoard Scalars

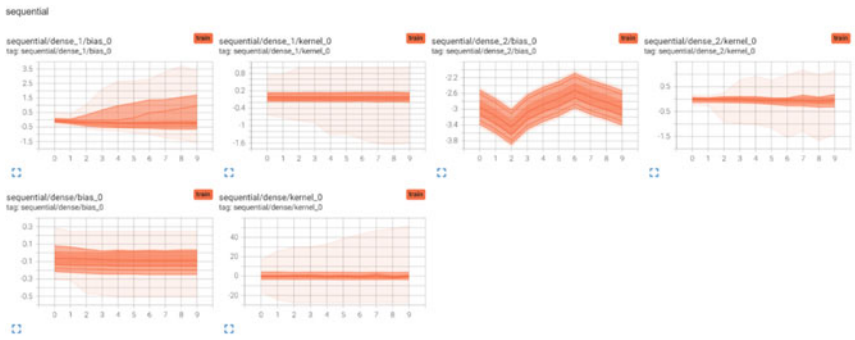

Fig. 16 AI Model performance analysis using TensorBoard Distributions

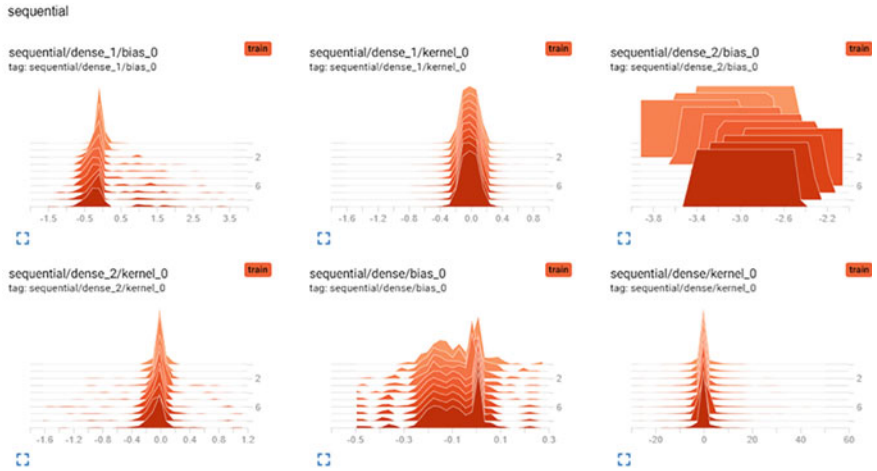

Fig. 17 AI Model performance analysis using TensorBoard Histogram

dataset needed more traffic flows with malicious traffic for the model to learn from and probably more variation in the attack types as well. A similar dataset IoT-23 has large number of traffic flow with the model able to learn from more than 745,000 flows of data with most of it being malicious. With large number of flows to test and validate the system, the AI model was able to achieve accuracies of over 99% while the accuracy was limited to just 46% for the VM dataset. A similar trend continues with the loss values where the VM dataset posted values of 0.69% while the IoT-23 dataset which features more columns of data, variations in data resulted in very little data loss value of 0.0093%.

The AI model proposed in this research will improve the effectiveness of traffic monitoring in a network and would suit the security requirements of SMEs. The model handles the process of network monitoring and can automatically take appropriate response actions relevant to a particular scenario. The traffic in a network will be continuously analysed by the AI model, and the anomalous traffic will be identified leading to an alert or appropriate action. The lack of dedicated security analysts is neutralised with the AI model's ability to handle automation of alerts and appropriate actions. The use of AI model answers the security challenges in an SME environment by placing emphasis on continuous monitoring and analysis for malicious patterns before a threat can be executed. By setting optimum threshold levels in malicious traffic resemblance with the trained data, a threat can be neutralised at the roots before transforming to a large-scale attack leading to business losses.

The performance metrics of 'accuracy' and 'loss' defines the model's suitability in an SME environment. With well-structured dataset an accurate AI model can be designed, and the performance of the current research model ascertains the same. The functions of ReLU (Rectified Linear Unit) and Adam optimizer can be adjusted to any network's requirement to achieve better results.

5 Conclusions and Future Work

The research aims to support and improve security standards in SMEs (Small and Medium Enterprise) by developing an artificial intelligence based malicious traffic detection system. SMEs often lack enough knowledge, infrastructure, and resources to implement a security system capable of actively analysing the network traffic for malicious activity. A system capable of automatically identifying threats and acting upon them simultaneously is unusual and out of bounds for SMEs, which is the motivating factor for the current research.

The project is set with a goal of developing an AI based tool capable of detecting threats in an SME network which is achieved by training an AI model to learn from training data consisting of attack traffic and regular traffic. Once the model has gained enough knowledge, it can analyse the network traffic for anomalies and provide alerts or appropriate response actions to mitigate the threat from becoming a reality.

The proposed AI model recorded an average performance with an accuracy level of 46% when tested using a dataset created in a virtual lab environment. To compare the model through various scenarios, further tests were carried out using alternate dataset, the IoT-23. The model performed exceptionally well with the evaluation parameters of accuracy achieving consistently more than 99% underlining the proposed model's performance. The sub-par performance with the primary dataset can be attributed to the lack of large number of malicious data flows within the VM dataset which limited the learning factor for the proposed model. The test using the IoT-23 dataset produced more commendable performance by topping the evaluation metrics of 99% accuracy and 0.0068% loss. The model indicates that the AI model provides the security boost necessary for SMEs to thwart any potential cyberattacks.

While the proposed model potentially brings invaluable augmenting capabilities in an SME environment, there are few hurdles that needs to be overcome to realize the true potential of AI system. The primary issue is with integrating the AI system to the current network infrastructure which might require additional infrastructure to aid the processing requirement of artificial intelligence. An effective integration mechanism of the AI model needs to be defined according to the network architecture with adequate security measures placed for the model itself as it processes every bit of information in the entire network. Another important factor would be the knowledge requirement to data pre-processing which directly affects the performance of the model. An incorrect feature extraction can lead to more damage as the system is based on the key features of the dataset. While there still exists some unknowns in the whole research due to the lack of real-time implementation, it is expected to provide additional security capabilities to an existing network architecture and could perfectly augment in making an informed decision about security incidents.

- Future Work

While the current model performed well in the experimental tests carried out in this research, the implementation in a real network is desirable which will be a part of future work. The AI model's learning patterns needs more analysis by running the

model for more iterations (100 epochs at the least). Future work will be a complete end-to-end fully integrated tool capable of analysing network traffic on the fly.

References

1. Abuadlla Y, Kvascev G, Gajin S, Jovanovic Z (2014) Flow-based anomaly intrusion detection system using two neural network stages. Comput Sci Inf Syst 11(2):601–622. Available at: http://www.doiserbia.nb.rs/img/doi/1820-0214/2014/1820-02141400035A.pdf. Accessed 21 Dec 2021
2. Ahmed A, Jabbar W, Sadiq A, Patel H (2020) Deep learning-based classification model for botnet attack detection. J Ambient Intell Hum Comput. https://doi.org/10.1007/s12652-020-01848-9. Accessed 18 Dec 2021
3. Aljabri M, Aljameel S, Mohammad R, Almotiri S, Mirza S, Anis F, Aboulnour M, Alomari D, Alhamed D, Altamimi H (2021) Intelligent techniques for detecting network attacks: review and research directions. Sensors 21(21):7070. https://doi.org/10.3390/s21217070. Accessed 17 Dec 2021
4. Al-Qatf M, Lasheng Y, Al-Habib M, Al-Sabahi K (2018) Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access 6:52843–52856. https://doi.org/10.1109/ACCESS.2018.2869577. Accessed 19 Dec 2021
5. Anderson M, Anderson S (2006) Guest editors' introduction: machine ethics. IEEE Intell Syst 21(4):10–11. https://doi.org/10.1109/MIS.2006.70. Accessed 28 Dec 2021
6. Beaver J, Symons C, Gillen R (2012) A learning system for discriminating variants of malicious network traffic. In: 8th Annual cyber security and information intelligence research workshop. Association for Computing Machinery, New York. https://doi.org/10.1145/2459976.2460003. Accessed 18 Dec 2021
7. Chamberlain L, Davis L, Stanley M, Gattoni B (2020) Automated decision systems for cyber-security and infrastructure security. In: 2020 IEEE security and privacy workshops (SPW). IEEE, San Francisco, pp 196–201. Available at: https://doi.org/10.1109/SPW50608.2020.00048. Accessed 18 Oct 2021
8. Chan L, Morgan I, Simon H, Alshabanat F, Ober D, Gentry J, Min D, Cao R (2019) Survey of AI in cybersecurity for information technology management. In: 2019 IEEE technology & engineering management conference (TEMSCON). IEEE, Atlanta, pp 1–8. https://doi.org/10.1109/TEMSCON.2019.8813605. Accessed 18 Oct 2021
9. Chou L, Tseng C, Lai M, Chen W, Chen K, Yen C, Ou T, Tsai W, Chiu Y (2018) Classification of malicious traffic using tensorflow machine learning. In: 2018 International conference on information and communication technology convergence (ICTC). IEEE, Jeju, pp 186–190. https://doi.org/10.1109/ICTC.2018.8539685. Accessed 19 Dec 2021
10. Dutta V, Choraś M, Pawlicki M, Kozik R (2020) A deep learning ensemble for network anomaly and cyber-attack detection. Sensors 20(16):4583. https://doi.org/10.3390/s20164583. Accessed 9 Dec 2021
11. Flammini F, Gaglione A, Mazzocca N, Pragliola C (2008) DETECT: a novel framework for the detection of attacks to critical infrastructures. In: Martorell et al (eds) Safety, reliability and risk analysis: theory, methods and applications, pp 105–112
12. García S, Grill M, Stiborek J, Zunino A (2014) An empirical comparison of botnet detection methods. Comput Secur 45:100–123. https://doi.org/10.1016/j.cose.2014.05.011. Accessed 25 Dec 2021
13. Hofstetter M, Riedl R, Gees T, Koumpis A, Schaberreiter T (2020) Applications of AI in cybersecurity. In: 2020 Second international conference on transdisciplinary AI (TransAI). IEEE, pp 138–141. https://doi.org/10.1109/TransAI49837.2020.00031. Accessed 18 Oct 2021
14. Jaigirdar F, Rudolph C, Oliver G, Watts D, Bain C (2020) What information is required for explainable AI?: a provenance-based research agenda and future challenges. In: 2020 IEEE

6th international conference on collaboration and internet computing (CIC). IEEE, Atlanta, pp 177–183. https://doi.org/10.1109/CIC50333.2020.00030. Accessed 20 Oct 2021
15. Johnson R (2019) 60 Percent of small companies close within 6 months of being hacked. Cybercrime Magazine. Available at: https://cybersecurityventures.com/60-percent-of-small-companies-close-within-6-months-of-being-hacked/. Accessed 9 Jan 2022
16. Kamoun F, Iqbal F, Esseghir M, Baker T (2020) AI and machine learning: a mixed blessing for cybersecurity. In: 2020 International symposium on networks, computers and communications (ISNCC). IEEE, Montreal. https://doi.org/10.1109/ISNCC49221.2020.9297323. Accessed 20 Oct 2021
17. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: 3rd International conference for learning representations. Cornell University, San Diego. Available at: https://arxiv.org/abs/1412.6980. Accessed 4 Jan 2022
18. Kurihara K, Katagishi K (2014) A simple detection method for DoS attacks based on IP packets entropy values. In: 2014 Ninth Asia joint conference on information security. IEEE, Wuhan. https://doi.org/10.1109/AsiaJCIS.2014.20. Accessed 25 Dec 2021
19. Kurpjuhn T (2019) Demystifying the role of AI for better network security. Network Secur 2019(8):14–17. Available at: https://www.sciencedirect.com/science/article/pii/S1353485819300972. Accessed 17 Oct 2021
20. Macas M, Wu C (2020) Review: deep learning methods for cybersecurity and intrusion detection systems. In: 2020 IEEE Latin-American conference on communications (LATINCOM). IEEE, Santo Domingo, pp 1–6. https://doi.org/10.1109/LATINCOM50620.2020.9282324. Accessed 18 Oct 2021
21. Marr B (2021) What is the importance of artificial intelligence (AI)|Bernard Marr. Bernard Marr. Available at: https://bernardmarr.com/what-is-the-importance-of-artificial-intelligence-ai/. Accessed 17 Oct 2021
22. Mesevage T (2021) What is data preprocessing & what are the steps involved? [Blog] MonkeyLearn. Available at: https://monkeylearn.com/blog/data-preprocessing/. Accessed 25 Dec 2021
23. Mohammad R, Alsmadi M (2021) Intrusion detection using Highest Wins feature selection algorithm. Neural Comput Appl 33(16):9805–9816. https://doi.org/10.1007/s00521-021-05745-w. Accessed 22 Dec 2021
24. Mohammad R, Thabtah F, McCluskey L (2013) Predicting phishing websites based on self-structuring neural network. Neural Comput Appl 25(2):443–458. https://doi.org/10.1007/s00521-013-1490-z. Accessed 17 Dec 2021
25. Müller V (2020) Ethics of artificial intelligence and robotics, 1st edn. Metaphysics Research Lab, Stanford University, Stanford
26. Nguyen K, Hoang D, Niyato D, Wang P, Nguyen D, Dutkiewicz E (2018) Cyberattack detection in mobile cloud computing: a deep learning approach. In: 2018 IEEE wireless communications and networking conference (WCNC). IEEE, Barcelona. https://doi.org/10.1109/WCNC.2018.8376973. Accessed 22 Dec 2021
27. Ouchchy L, Coin A, Dubljević V (2020) AI in the headlines: the portrayal of the ethical issues of artificial intelligence in the media. AI Soc 35(4):927–936. https://doi.org/10.1007/s00146-020-00965-5. Accessed 28 Dec 2021
28. Pelley S (2019) Facial and emotional recognition; how one man is advancing artificial intelligence. Cbsnews.com. Available at: https://www.cbsnews.com/news/60-minutes-ai-facial-and-emotional-recognition-how-one-man-is-advancing-artificial-intelligence/. Accessed 17 Oct 2021
29. Press G (2016) Cleaning big data: most time-consuming, least enjoyable data science task, survey says. Forbes. Available at: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#58fdfc6f637d. Accessed 25 Dec 2021
30. Sapavath N, Muhati E, Rawat D (2021) Prediction and detection of cyberattacks using AI model in virtualized wireless networks. In: 2021 8th IEEE international conference on cyber security and cloud computing (CSCloud)/2021 7th IEEE international conference on edge computing

and scalable cloud (EdgeCom). IEEE, Washington, DC, pp 97–102. https://doi.org/10.1109/CSCloud-EdgeCom52276.2021.00027. Accessed 19 Oct 2021
31. Shaikh R (2018) Feature selection techniques in machine learning with python. Medium. Available at: https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e. Accessed 7 Jan 2022
32. Svenkatsai123 (2021) Why tensorflow is so popular—tensorflow features. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/why-tensorflow-is-so-popular-tensorflow-features/?ref=rp. Accessed 25 Dec 2021
33. Thaseen I, Poorva B, Ushasree P (2021) Network intrusion detection using machine learning techniques. In: 2020 International conference on emerging trends in information technology and engineering (ic-ETITE). IEEE, Vellore. https://doi.org/10.1109/ic-ETITE47903.2020.148. Accessed 21 Dec 2021
34. Wu P, Guo H (2021) LuNet: a deep neural network for network intrusion detection. In: 2019 IEEE symposium series on computational intelligence (SSCI). IEEE, Xiamen, pp 617–624. https://doi.org/10.1109/SSCI44817.2019.9003126. Accessed 19 Dec 2021
35. Yuan X, Li C, Li X (2017) DeepDefense: identifying DDoS attack via deep learning. In: 2017 IEEE international conference on smart computing (SMARTCOMP). IEEE, Hong Kong, pp 1–8. https://doi.org/10.1109/SMARTCOMP.2017.7946998. Accessed 17 Dec 2021
36. Zeadally S, Adi E, Baig Z, Khan I (2020) Harnessing artificial intelligence capabilities to improve cybersecurity. IEEE Access 8:23817–23837. https://doi.org/10.1109/ACCESS.2020.2968045. Accessed 18 Oct 2021