DS Journal of Cyber Security
Volume 1 Issue 1, 1-18, Jul - Sep 2023
ISSN: 2584-0665 / https://doi.org/10.59232/CYS-V1I1P101

Original Article

Machine Learning for Improving the Security of Mobile Devices against Spyware: The Case of Pegasus

Patrick Dany Bavoua Kenfack[1], Alphonse Binele Abana[2], Emmanuel Tonye[3], Armel Brice Djousse Nguegang[4]

[1,2,3]*Department of Electrical and Telecommunications Engineering, National Advanced Polytechnic School, University of Yaounde I, Centre, Cameroon*
[4]*Ecole Supérieure d'Ingénieurs Numérique et Matériaux, Dijon, France*

[1]*danybavoua@gmail.com*

Received: 16 May 2023; Revised: 23 May 2023; Accepted: 5 June 2023; Published: 6 July 2023;

Abstract - Given the different approaches developed in the literature to guarantee the confidentiality of user data in smartphones, and the attack approach which is currently associated with the contemporary Spyware giant Pegasus, we are carrying out in this work a reflection on the development of a protection tool based on machine learning. Focused on 0-day and even 0-click type approaches, the behaviour of Pegasus leads us to choose to carry out an IDS based on detection by behavioural analysis. With the challenges of considering the reduced performance of critical resources of Android phones, the proposed system uses the auto encoder algorithm, applied to a network of multilayer and back-propagation perceptrons. For the construction of the model, the search for a robust structure is done with the help of the KDD+aggregate dataset, and the construction of the model with the IA-AE-IDS set that we have processed for the occasion. The model obtained with precision and a recall of at least 98% is then deployed on the open-source network analysis application PCAPdroid, mainly in the form of a service that the user can start and stop at will.

Keywords - Spyware, Machine learning, Pegasus, Auto encoder, Cyber security

1. Introduction

Mobile terminals are the most fashionable in a progressively digitized environment and with the massive adoption of telecommunications tools. These are various portable devices allowing the exchange and processing of data. Having become essential in the functioning of organizations and forming part of the fundamental components of business processes, they can be categorized into three classes. The fixed mobile terminals present in locomotives, like the screens for GPS in the vehicles, the portable mobile terminals whose mobility is associated with that of a person or a machine like the fixed telephones, and the portable mobile terminals, which are entirely autonomous like smartphones.

This third category of mobile terminals seems to be the most popular nowadays and has been experiencing for some time, with the second category, an exponential growth due to the advent and explosion of the Internet of Things (IoT). Indeed, in a world where everything wants to be connected, we encounter increasingly more portable terminals, such as connected watches, and portable terminals, such as household appliances. In this wake, where intelligent grids, smart cities, and smart homes are mentioned, the innovative concept highlights the connectivity of the elements of a defined system. In this panoply of tools, mobile phones and computers make it possible to caricature their owners better because they are used continuously.

Mobile phones prevail in this sector because of their portability. It is, therefore, a proven point of vulnerability for spying on owners. These perpetually encounter more and more varied and harmful threats. Aware of these realities, several basic countermeasures are defined within the different devices without always being up to the attack.

In addition, the software is developed daily to ensure users' safety further. These include, for example, anti-viruses such as Kaspersky, Symantec, Avast, etc. Other modules like IDS and IPS are also set up for defensive purposes. However, the production of such generalist tools remains difficult because their structures depend on the target equipment, the category of threat considered, and the interview method. The development of countermeasures for handheld mobile phones in its evolution includes new proposals and updates to existing solutions. Depending on whether the phone runs on android, iOS or any other Operating System, a wide range of products answers the question of system security against espionage. However, the Android system remains the most insecure OS, with around 95% of attacks targeting it. This is one of the reasons why the concern remains constant and inflected. Moreover, the reflections, in addition to being concerned with the question of insecurity, also dwell on the limited resources of this device category.

Before the rise of AI, primarily static solutions were signature-based. The objective is to continuously detect and recognize new attack signatures and save them in a database to exploit them for future detection. However, this method is resource-intensive and ineffective in the face of new attacks, especially in this climate where the proliferation of various attacks litter uses. Dynamic methods and anomaly detection techniques have therefore emerged. Their approaches involve learning normal behaviour profiles by extracting a so-called normal model from a data set obtained from the system. The advent of AI has further galvanized these methods, which have the added value of detecting unknown attacks. However, approaches based on Machine Learning (ML) need a lot of computing power, and those based on Data-Mining (DM) require a large amount of data and, therefore, space. These dynamic methods are deployed on the host or a remote server [1].

Our work will therefore consist, after having presented the contextual framework of this work for a while, of answering the question of which resolution approach based on intrusion detection models using artificial intelligence could be proposed against Spyware defined and acting according to the execution structure of the Pegasus Spyware of the Israelite NSO structure.

Our overall goal in this study is to provide a framework for combating Spyware on Android, focusing on Pegasus, which is currently in the news.

2. Mobile Device Security and Threats

Mobile device security is done at many levels, like the core, software platform, and application levels [1]. We will be interested in this section on the threats by Spyware.

2.1. Spying

Espionage is a discreet and sneaky activity, consisting of monitoring a third party (a victim) in his actions and words to make a report and profit. It is undoubtedly possible physically, but technology has given it a different face. If proximity was required in the first case, it is unnecessary in the second case. Indeed we are witnessing increasingly more remote espionage through computer equipment: it is computer espionage.

Computer systems are very generally clusters of vulnerabilities or flaws. The likelihood of people (attackers) impacting the operation of a system through its vulnerabilities also poses a threat to these systems and their users. However, these cyber agents must carry out specific active and/or passive attacks aimed at the targeted system to achieve their ends. It's attacks include sniffing, man in the middle, DNS/DHCP Spoofing, simple or distributed Denial of Service (DOS and DDOS), phishing (or phishing) and spear phishing, SQL injection, XSS, password or malware attacks, etc. [2, 3]. The objective sought varies according to the attack carried out. However, a distinction is made between network congestion, the unavailability of the system or part of it, its decommissioning, or infiltration.

The latter, under the name of intrusion in computer jargon, is the result of the success of an exploit, which is a kind of attack like phishing, backdoor, Trojan, buffer overflow, SQL injection, cross site scripting or XSS, the purpose of which is to take advantage of vulnerabilities in a system and its components, in order to grant access and control illicitly. Exploits usually come in the form of software (Spyware or Spyware) or a piece of code and target either the interconnections (the network) or the components (the terminals/hosts). A payload then makes it possible to achieve the targeted objectives, which in this case are either the illicit insertion of information into the system, the harming of the system, or then, as is very often the case, the illicit recovery of information: espionage [22, 23].

2.2. *Spywares*

Spyware inherits from espionage the properties of stealth and sneakiness. Therefore, such software must be invisible to the user and intrusion detection software. To do this, several rootkits or concealment methods are used, such as repackaging, Trojan horses, and stealth mode (not visible in running processes). The types of intrusions generally encountered are phishing, backdoor, Trojan horse, and social engineering. Among the intrusion vectors encountered, it can be listed: protection cracking software such as cracks and keygens (shareware), specific free software (freeware), fake security software (rogues), dubious websites, web messaging, and SMS [4, 5].

It is now appropriate to concretely browse some of this software. On the one hand, it should be noted that while some are legally and officially available and usable, other legitimate remains; on the other hand, any software/application not acquired by official means has a probability of being Spyware.

2.3. *Pegasus*

From the family of Spyware or Spyware, Pegasus is a work of one of the "NSO Group". It is an Israeli computer security company founded in 2010 by Niv Carmi, Shalev Hulio and Omri Lavie and located in Herzliya, Israel [1]. Their objective is to manufacture and market state-of-the-art equipment intended to fight against terrorism and organized crime. Owner of the Pegasus spyware, which it also sells to state organizations under the approval of the Israeli Ministry of Defense [2], the company has for several years been in the grip of an ethical controversy over the use of this latest product. The controversy was raised following work jointly by Amnesty International and the consortium of journalists Forbidden Stories since 2019.

Indeed, Pegasus is a Spyware targeting iOS and android smartphones. Its attacks are based on phone numbers and therefore have an international reach. They are of four orders [3]:

- Zero-day attack: Based on vulnerabilities, i.e. unknown to the software publisher. Generally exploiting zero-click flaws, i.e. requiring no user action to infect the equipment. For example, it would suffice to make a call (by normal means or by VoIP) or to send an SMS to a target device to infect it, even if the call has not been picked up and/or the message opened. Everything is transparent to the user.
- Attack by spear phishing: From the large family of phishing-type attacks, spear phishing is a variant of phishing bred by social engineering. Requiring in the case of Pegasus, where the user clicks on a link delivered by message (SMS or iMessage), the Spyware has been installed thanks to software flaws detected within the system.

- Attack by network injection (or by the middle man): Acting at the network level, the goal is to discreetly position itself between the victim and his router to capture all the traffic passing through there. According to the NGO forbidden stories, this method is used by Pegasus to prepare the ground. Thus [4], by hijacking the network, pegasus leads the victim device to download to install the Spyware without its use being notified.
- Wireless Transceiver Attack: "Where neither spear-phishing nor clickless attacks succeed, Pegasus can also be installed on a wireless transceiver located near a target, or, according to an NSO brochure, simply installed manually if an agent can steal the target's phone" [5].

The attack modes erecting the functioning of Pegasus depend on the version used. Two injection methods are thus known to date. As for the first version of the software, to be infected, the target had to click on a link which was to be responsible for starting the execution of the PEGASUS installation processes on the targeted device. As for the new versions, they rely on software ZERO-DAY flaws using ZERO-CLICK attacks.

Once installed, the software sends the data back to the sponsor, creating a tunnel. The spy will be able to take control through the software:

- Network setting data;
- Text, audio, and video communications from messaging apps and social networks are intercepted; of the WEBCAM, etc.
- The particularity of PEGASUS is, above all, its great discretion and its ability to self-destruct, which makes the software difficult to detect in some instances.

Regarding how the attack unfolded, CITIZENLAB established that the PEGASUS system relies on three main components: a workstation, an infection server, and a cloud infrastructure.

The operator launches his attack from his station, which causes the sending of the trapped SMS. The link incorporates points to one of the cloud infrastructure web servers. The WEB server then redirects the victim to the infection server, which will execute the attack, as shown in Figure 1.

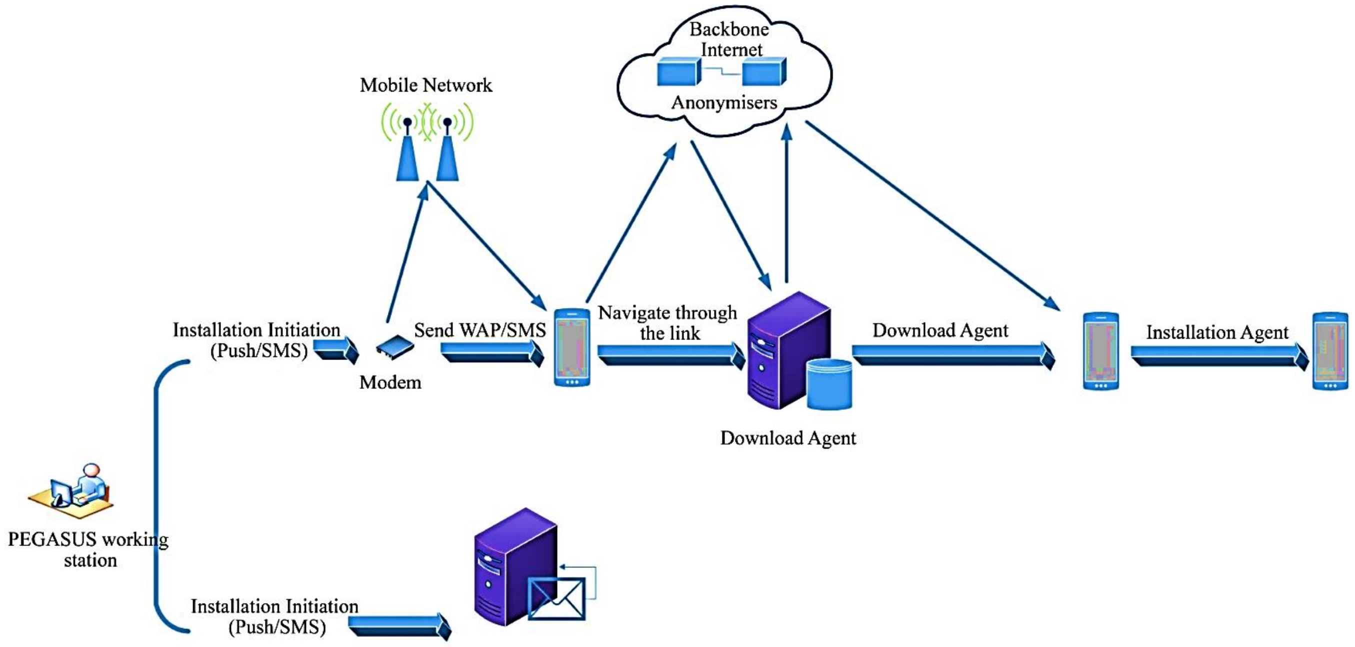


Fig. 1 Diagram of an attack carried out by PEGASUS

2.4. The Contribution of Artificial Intelligence in the Fight against Spyware

We have previously noticed that the basic security of Operating Systems seems insufficient in the face of the evolution of threats and implementation processes. So talking about anti-spyware refers to discussing additional detection techniques and countermeasures. Furthermore, from all the above, it is clear that they sport several forms, depending on their modes of operation. Therefore, one of the main security activities is to work against Spyware through intrusion detection, which is "the process of observing the behaviour of the system to identify inappropriate activity" [6]. Two approaches make it possible to create these Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS). These are the signature method (misuse detection) and the anomaly or behavioural method (anomaly detection). These apply either for detection on the network Network-bases IDS (NIDS) or for detection on the host for Host-bases IDS (HIDS) [8, 9, 21].

2.4.1. Signature-Based Intrusion Detection

The signature here is meant to be the identifying character. It can be either a byte sequence or an execution pattern. In the first case, the method based on signatures will be a static approach, while in the second case, it will be a dynamic approach [14, 15].

The static approach is based on the fact that each malware and Spyware contains a fingerprint that characterizes it. The detection operation, therefore, consists of looking for the presence of a known pattern. This research is based on a signature database progressively updated thanks to previous attacks listed and exploited. This is particularly the approach adopted by almost all anti-viruses today. However, if this approach works with viruses, it seems evident that it will not be the case with Trojan horses, which will have no harmful appearance in their appearance and attacks with unlisted signatures [16, 19].

The dynamic approach is, therefore, a prospect for this non-negligible category. Indeed, this approach relies on execution information. In this case, it is a question of determining a sequence of states listed as dangerous. This is how this model makes it possible to bring enough abstraction to this method to detect unknown attacks but which part of an already recognized class are of attacks [7, 10, 11].

Building an Anomaly Detection System (ADS) is a two-step process: learning and detecting. The learning phase consists of building a profile that will be qualified as usual. This is done by observing and analyzing the monitored entity. The detection phase is the period during which the IDS evaluates the similarity ratio between the behaviour of the system and the profile in order to detect and report possible attacks and/or instructions. It is a question of signalling an anomaly in the system, although any anomaly is not an intrusion or an attack. Already it could have more or less false positives depending on whether the expected behaviour is correct and/or complete. In addition, depending on the learning procedure of the IDS, there is also in this behavioural method a static approach and a dynamic approach.

According to Frédéric Majorczyk, in the static approach, "the profile is established by observing the value of certain system parameters considered random variables. A statistical model is used for each system parameter to establish the distribution of the corresponding random variable. Once the model is established, a distance vector is calculated between the flow of observed events and the profile. If the distance exceeds a certain threshold, an alert is issued" [7, 12]. IDES (Intrusion Detection Expert System), proposed by Anderson in 1998, uses this approach by considering indices such as processor time, session duration, number of connection attempts, etc.

The dynamic approach is based on the ability of the system to extract from a flow of information the characteristics making it possible to define the profile. The feature extraction canvases are determined by experts or by Machine Learning (ML) or Data Mining (DM) techniques, which are techniques promoted by Artificial Intelligence (AI) [23, 24, 25].

This analytical study allows us to understand that the evolution of threats requires that the basic security of the Android portable terminal be supplemented with modules such as IDS, which can be nested at different levels depending on the desired result. Having also determined that Pegasus is recognized for information theft, we can now focus on techniques using AI processes in IDS, particularly ADS, to fight against information theft [13, 17].

3. Method, Material and Tools

3.1. Detection Method

In the account of this work, we propose intrusion detection on android phones, a modular framework based on behavioural analysis. This aims to identify normal behaviour concerning the exfiltration of sensitive data and then alert the user when an anomaly is detected. Indeed, frivolous to 0-days and 0-click type flaws, our hypothesis states that an analysis based on the communication profile would be more promising than one based on the system's functioning.

The proposed system is based on data flow analysis algorithms in an environment. For each category of sensitive data, the pattern strives, based on the chosen detection algorithm, to build a framework of standard patterns in a given period. This choice assumes the system's functioning can be expected for at least its first two weeks. Therefore, patterns of subsequent extraction activities of sensitive data are monitored against a comparison to the usual pattern. The detection of an anomaly prompts an alert. Everything being local to the telephone, the module is thus intended to be a detector of unauthorized communication. It is, therefore, a NIDS.

The system in this perspective is delimited in a field containing three main modules, as shown schematically in the capture of Figure 2.

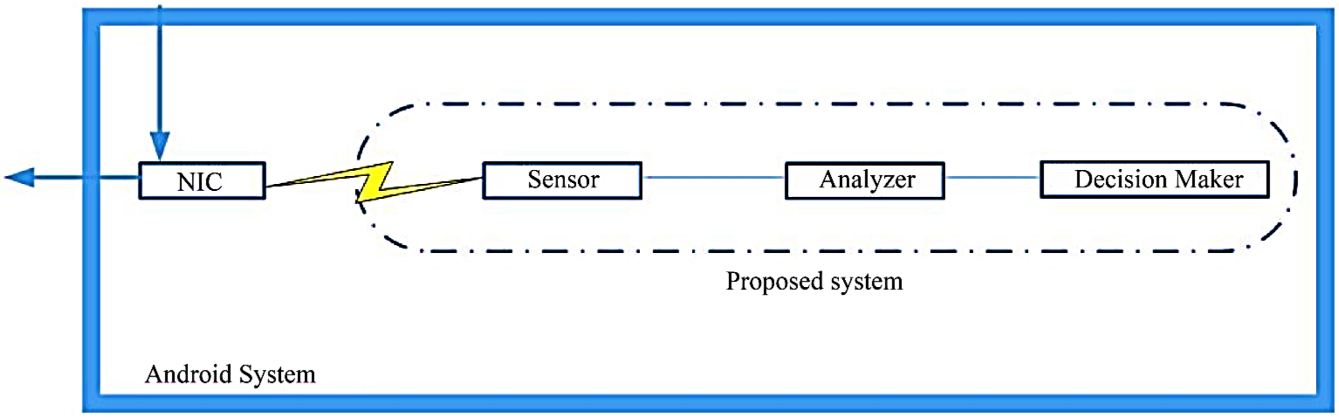


Fig. 2 Structure of the proposed solution

It is important to observe each module more closely in order to discern the challenges associated with it.

3.1.1. The Sensor

The sensor is a probe, so the goal is to collect the information the system will need. The collection of its data must be meticulous and well approached because they are used at two levels, on the one hand for the training of the model and on the other hand during the production period. Three solutions were evaluated in turn:

- The use of datasets, according to Table 1, made available by the laboratories to assist research in computer security. We have studied some of them.

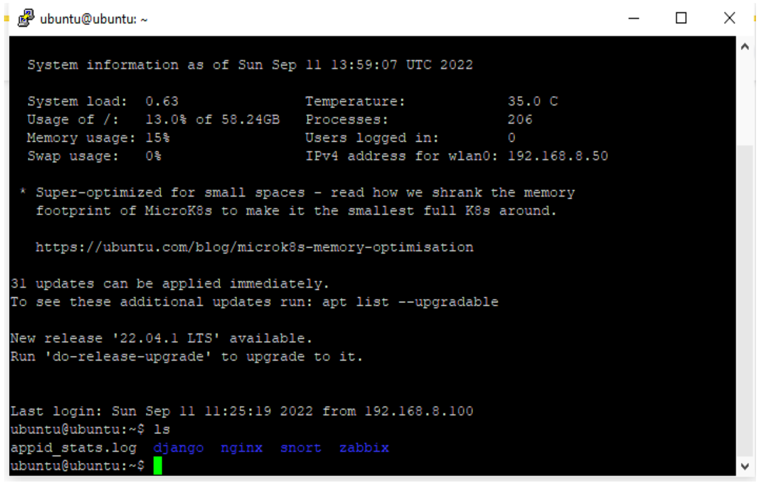
Table 1. Datasets used

Datasets	Difficulties
KDD, KDD99, KDD+aggregate, DARPA, CICDataset, etc.	Inability to address user profiling concerns.

- Using snort IDS in sniffing mode on the local network to harvest traffic related to the target phone. Table 2 and Figure 3 summarize the work carried out for this phase.

Table 2. Equipment used for IDS snort

Material	Characteristics	Software	Difficulties
Raspberry Pi 4 (Remote access by SSH with an HP brand laptop)	CPU: armvl7 RAM: 8GB SD-Card: 64GB	OS: ubuntu-20.04.4-preinstalled-server-armhf+raspi.img.xz Server: Snort 3.0	Packet processing to extract essential features for analysis.



```
 System information as of Sun Sep 11 13:59:07 UTC 2022

 System load:  0.63               Temperature:           35.0 C
 Usage of /:   13.0% of 58.24GB   Processes:             206
 Memory usage: 15%                Users logged in:       0
 Swap usage:   0%                 IPv4 address for wlan0: 192.168.8.50

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

31 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sun Sep 11 11:25:19 2022 from 192.168.8.100
ubuntu@ubuntu:~$ ls
appid_stats.log  django  nginx  snort  zabbix
ubuntu@ubuntu:~$
```

Fig. 3 Raspberry Pi4 and attempt to harvest data from snort

- The use of the Wireshark tool to obtain the data. This approach is summarized in Table 3.

Table 3. Hardware used by the Wireshark tool

Material	Method	Difficulties
Laptop HP	Positioning in MITM mode	Characteristics extracted from the packages are not satisfactory for achieving the objective.

3.1.2. The Analyzer

In the same way as the previous part, dealing with any element involved in an Artificial Intelligence mechanism requires thinking about the two sine-qua-nones phases, training and production. In addition, downstream, there are decisive operations for the success of these two phases, namely data processing.

- Pre-processing or data processing: Talking about data processing comes down to raising two essential elements, which are the adoption of a data format and the choice of characteristics. Indeed, during the training and prediction phases, the processed data has a format that depends on the chosen algorithm. We will return to this algorithm notion in the following subsection.

- Training or checking: As part of the training, the challenge is to find both a structure for the model and a combination of hyperparameters, to obtain better results after choosing the algorithm to use. Several metrics can be evaluated depending on the data type and the objective. Among others: Accuracy, which is the correct prediction rate; Sensitivity, which is the correct positive prediction rate; Specificity, which is the rate of good negative prediction; Precision, which is the probability that a prediction on a category is correct.

3.1.3. The Decision Maker

The decision-maker module is the body responsible for announcing or not the presence of an anomaly in the system. Indeed, depending on the parser's output, the system should be able to signal to the user the presence of an unrecognized behaviour.

3.2. Anomaly Detection Algorithms: The Auto Encoder

Anomaly Detection System (ADS) includes many algorithms grouped into statistical methods, clustering methods, Machine Learning (ML), and many others. Their roles are to allow the training of a model and its use for predictions, classification or data analysis [1].

According to the comparative study of anomaly detection methods conducted by [9], we can note that the Deep Learning approach has a place of choice in detecting intrusion by behavioural analysis concerning data flows. Thus, we have chosen the Deep Learning category algorithm named Auto Encoder (AE). This one is based on the multilayer perceptrons [1, 20].

According to Figure 4, a self-encoding neural network is an unsupervised machine learning algorithm that applies back-propagation and aims to ensure that the reconstructed output values are closest to those received as input. The goal of an auto encoder is to learn a representation (encoding) of a set of data, usually to reduce the dimensionality of that set. It comprises three components, the encoder, the code and the decoder.

- Encoder: part of the perceptron network compresses input data into a representative latent space. The goal is to clean the data received by reducing noise as much as possible.
- Latent space: represents the part of the network that is supposed to have the quintessence of information: the code.
- Decoder: is the portion of the network whose purpose is to reconstruct the initial data from its code.

Training the model, therefore, consists of forming a structure capable of reconstructing the initial data from the code with the least possible error. In addition, there are several types of auto encoder:

- The convolutional auto encoder is an auto encoder that uses convolution operations to learn how to encode inputs, get their code, and reconstruct those. It is used for image reconstruction, colouring, etc.
- The deep auto encoder: each layer learns a main characteristic of the data. All together makes it possible to have a vital characteristic of the dataset.
- The contractive auto encoder is an unsupervised Deep Learning algorithm that helps the neural network encode unlabeled data. In this algorithm, similar data are contracted into a similar output materialized by a neighbourhood formed from the observations during the training phase.

3.3. Summary of Method and Tools Used

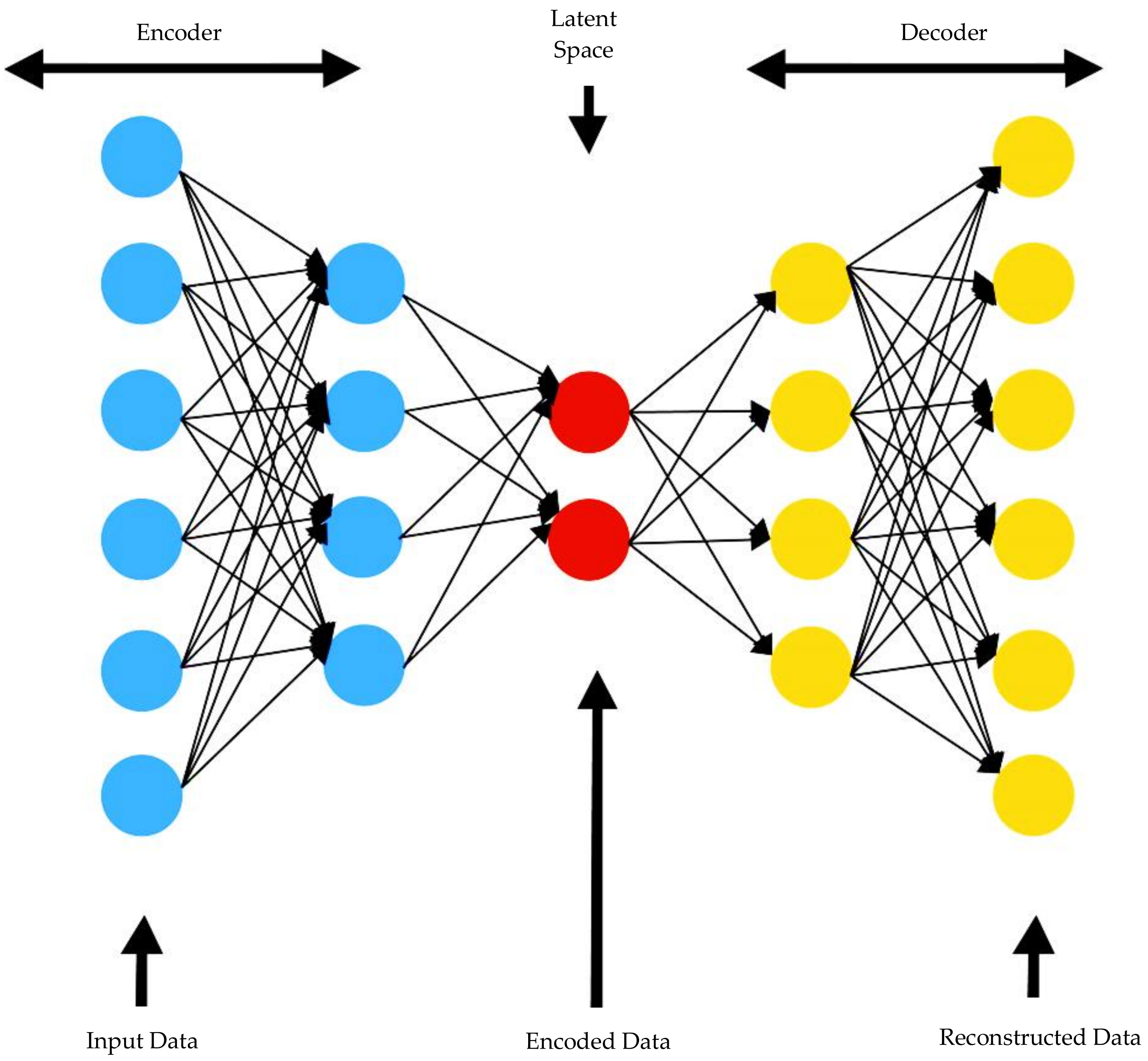


Fig. 4 Methodology of an auto-encoding neural network

By way of summary, we can retain that in the process of finding a solution approach, after reflecting on several models, it was decided that the solution to be implemented could be summarized in these steps:

- The use of the KDD+aggregate dataset for training the model using the auto encoder algorithm on a multilayer perceptron network. This is only to obtain a suitable architecture.
- The use of an IA-AE-IDS dataset (Personal dataset built based on an Android phone.) to train the model to be deployed.
- The use of the PCAPdroid open-source project for the deployment of the artificial intelligence solution.

The material and tools used can be summarised, as seen in [1].

4. Simulations, Results and Comments

Our application is based on the Multi Layers Perceptrons (MLP) model of the artificial neural network to make a supervised classification of the TCP/IP connections of the KDD+aggregate database in order to establish an intrusion detection system based on the analysis of the behaviour of these connections and allows them to be classified into two types (attack and regular).

4.1. Description of the KDD+aggregate Training Database

KDD99 is a dataset built in the 1998s by Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), then MIT Lincoln Labs8 collected and distributed the datasets to evaluate the computer network intrusion detection system. The Network Security Layer-Knowledge Discovery in Databases (NSL-KDD) was founded on the KDD99 dataset in 2010 by reducing redundancy and double recordings while seeking to maintain a certain balance between the different types of data. KDD+aggregate is subsequently an aggregation of the NSL-KDD database with other additional data.

4.2. Data Processing and Distribution

Regarding the data distribution, the dataset contains five (05) data classes. However, in our work, we want to distinguish what is expected from what is not, so we will work with two data classes. Tables 4, 5, and 6 show an evolving summary of the data sets.

Table 4. Summary table of the initial classes of KDD+aggregate

Class	**Normal**	**Probe**	**DOS**	**R2L**	**U2R**	**Total**
Number	67343	11656	45927	995	52	125973
Percentage (%)	53.46	9.25	36.46	0.8	0.041	100.011

Table 5. Summary table of the KDD+aggregate dataset under two classes

Class	**Normal**	**Abnormal**	**Total**
Number	67343	58630	125973
Percentage (%)	53.46	46.54	100

Table 6. Summary table of data used for training with KDD+aggregate

Categories	**Training data (80%)**	**Test data (20%)**	**Total (100%)**
Normal	53875	13468	67343
Abnormal	46904	11726	58630

4.3. Finding the Structure and Hyperparameters for the Model

The search for the structure and the hyper-parameter uses the KDD+aggregate dataset according to the flowchart defined in Figure 5. The different processing steps: data digitalization, data normalization, and selection of the best attributes, are described in [1].

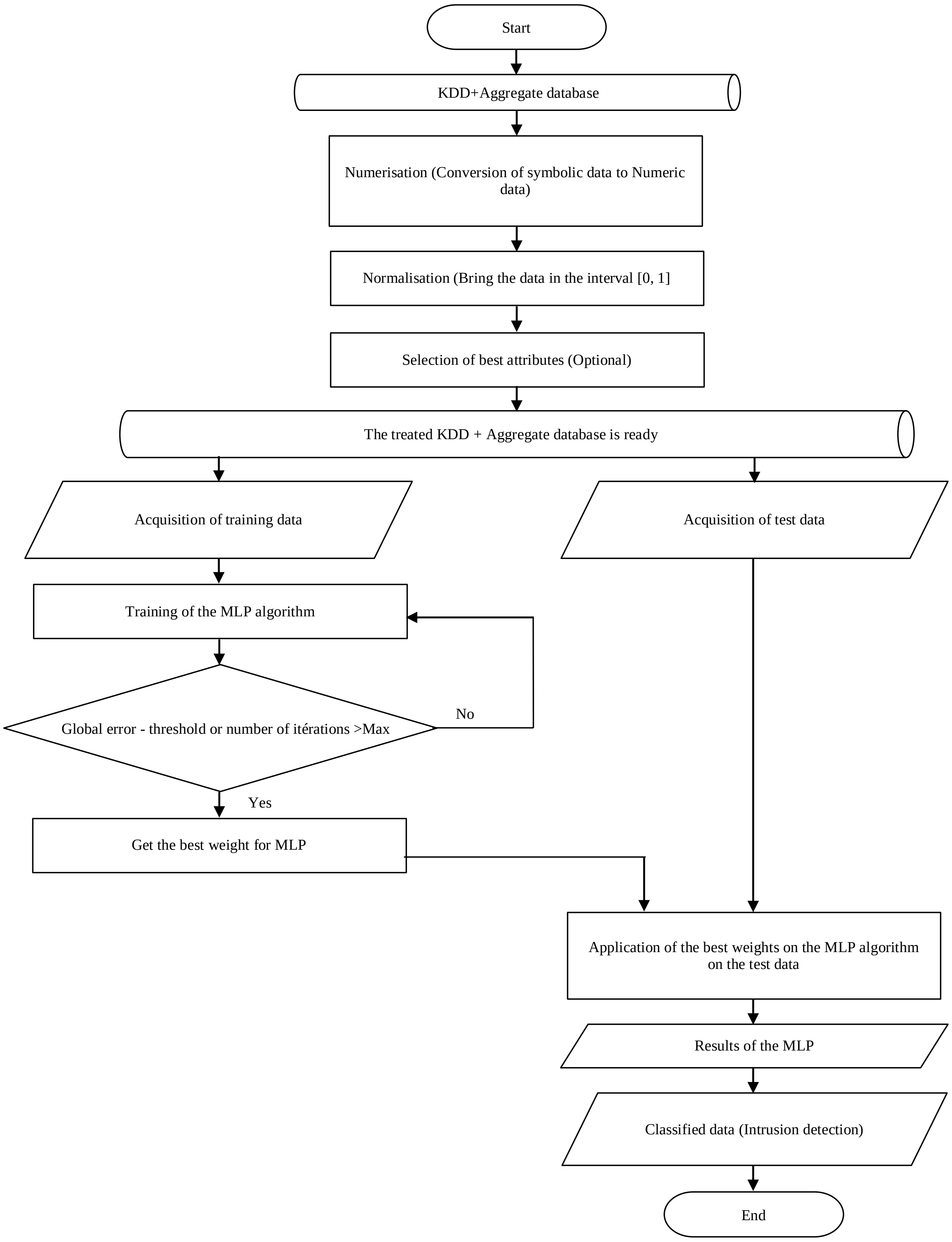
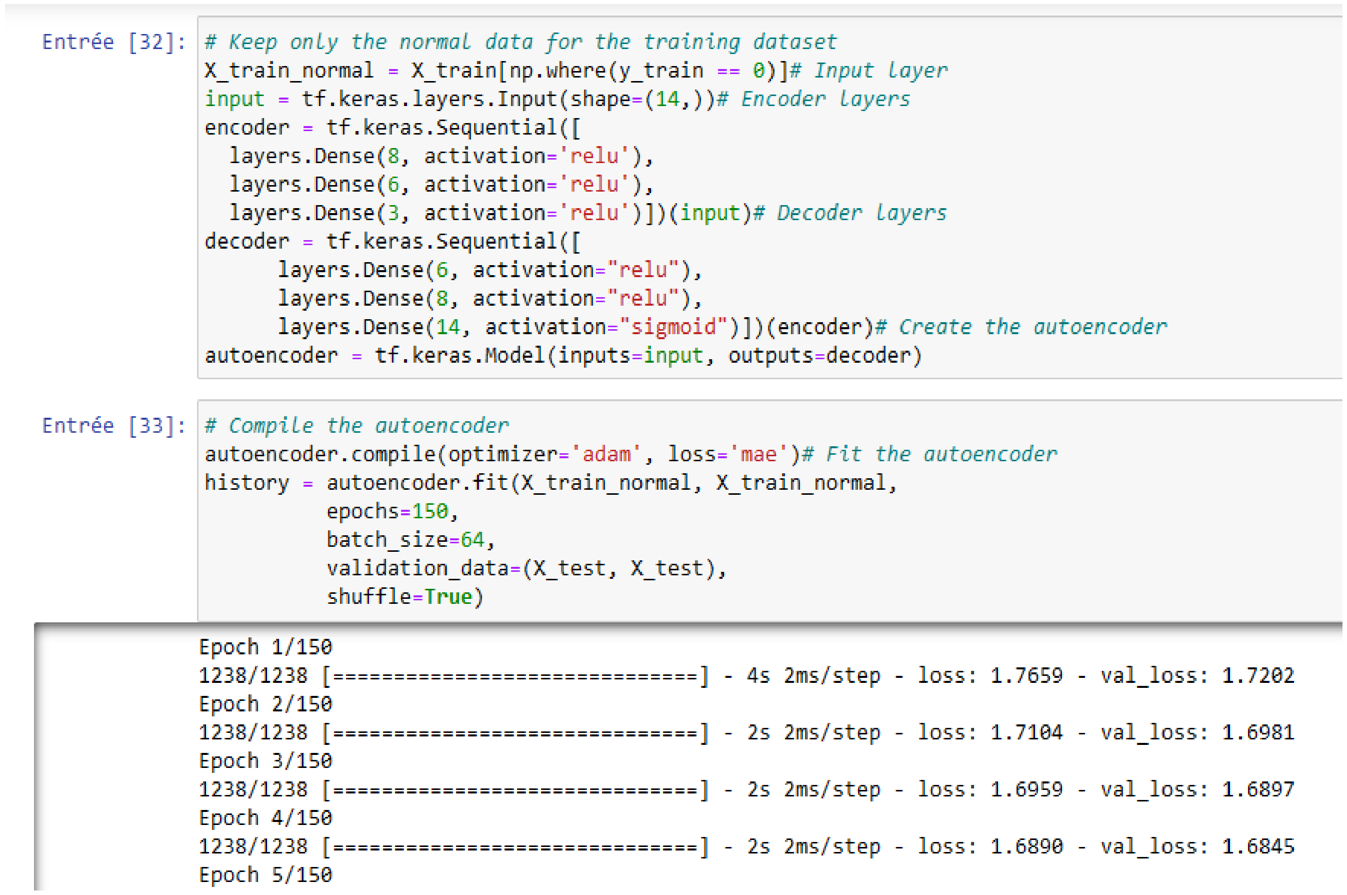

Start
KDD+Aggregate database
Numerisation (Conversion of symbolic data to Numeric data)
Normalisation (Bring the data in the interval [0, 1]
Selection of best attributes (Optional)
The treated KDD + Aggregate database is ready
Acquisition of training data
Acquisition of test data
Training of the MLP algorithm
Global error - threshold or number of itérations >Max
No
Yes
Get the best weight for MLP
Application of the best weights on the MLP algorithm on the test data
Results of the MLP
Classified data (Intrusion detection)
End

Fig. 5 Flowchart for creating the intrusion detection model

4.3.1. Finding Features for the Model

In this research, we trained several models and compared their metrics. After several tests, we retained a structure of seven (07) layers, as presented in Figures 6, 7 and 8.



```
Entrée [32]: # Keep only the normal data for the training dataset
             X_train_normal = X_train[np.where(y_train == 0)]# Input Layer
             input = tf.keras.layers.Input(shape=(14,))# Encoder layers
             encoder = tf.keras.Sequential([
               layers.Dense(8, activation='relu'),
               layers.Dense(6, activation='relu'),
               layers.Dense(3, activation='relu')])(input)# Decoder layers
             decoder = tf.keras.Sequential([
                   layers.Dense(6, activation="relu"),
                   layers.Dense(8, activation="relu"),
                   layers.Dense(14, activation="sigmoid")])(encoder)# Create the autoencoder
             autoencoder = tf.keras.Model(inputs=input, outputs=decoder)

Entrée [33]: # Compile the autoencoder
             autoencoder.compile(optimizer='adam', loss='mae')# Fit the autoencoder
             history = autoencoder.fit(X_train_normal, X_train_normal,
                       epochs=150,
                       batch_size=64,
                       validation_data=(X_test, X_test),
                       shuffle=True)

             Epoch 1/150
             1238/1238 [==============================] - 4s 2ms/step - loss: 1.7659 - val_loss: 1.7202
             Epoch 2/150
             1238/1238 [==============================] - 2s 2ms/step - loss: 1.7104 - val_loss: 1.6981
             Epoch 3/150
             1238/1238 [==============================] - 2s 2ms/step - loss: 1.6959 - val_loss: 1.6897
             Epoch 4/150
             1238/1238 [==============================] - 2s 2ms/step - loss: 1.6890 - val_loss: 1.6845
             Epoch 5/150
```

Fig. 6 Finding a structure for the model

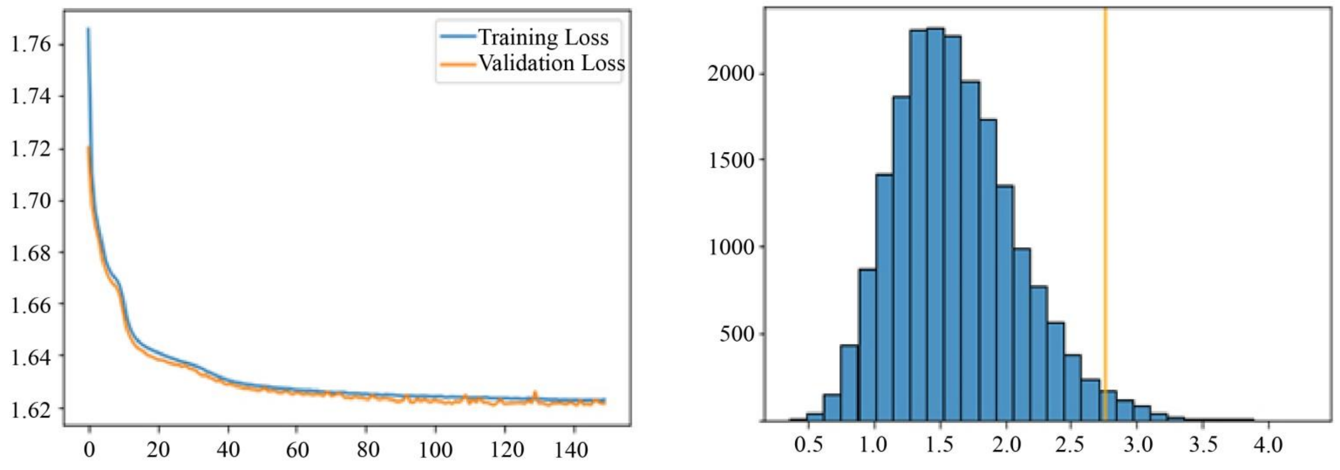
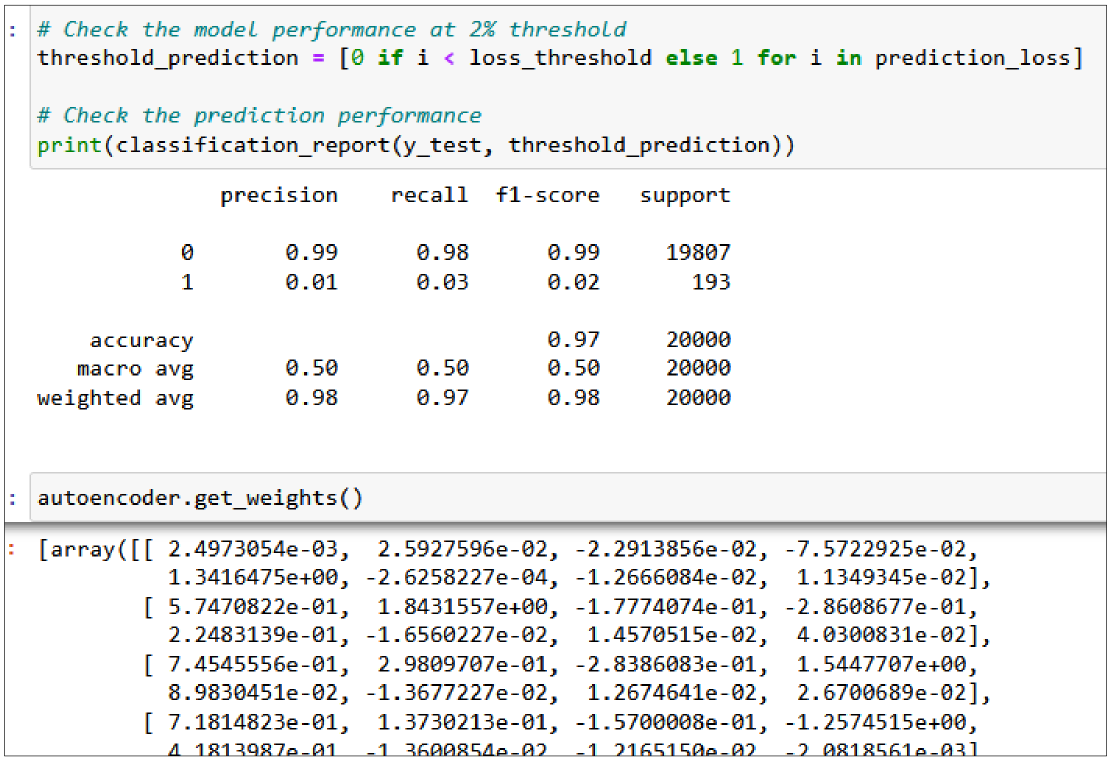


Fig. 7 Training error curve and data distribution



```
# Check the model performance at 2% threshold
threshold_prediction = [0 if i < loss_threshold else 1 for i in prediction_loss]

# Check the prediction performance
print(classification_report(y_test, threshold_prediction))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99     19807
           1       0.01      0.03      0.02       193

    accuracy                           0.97     20000
   macro avg       0.50      0.50      0.50     20000
weighted avg       0.98      0.97      0.98     20000
```

```
autoencoder.get_weights()
```

```
[array([[ 2.4973054e-03,  2.5927596e-02, -2.2913856e-02, -7.5722925e-02,
          1.3416475e+00, -2.6258227e-04, -1.2666084e-02,  1.1349345e-02],
        [ 5.7470822e-01,  1.8431557e+00, -1.7774074e-01, -2.8608677e-01,
          2.2483139e-01, -1.6560227e-02,  1.4570515e-02,  4.0300831e-02],
        [ 7.4545556e-01,  2.9809707e-01, -2.8386083e-01,  1.5447707e+00,
          8.9830451e-02, -1.3677227e-02,  1.2674641e-02,  2.6700689e-02],
        [ 7.1814823e-01,  1.3730213e-01, -1.5700008e-01, -1.2574515e+00,
          4.1813987e-01, -1.3600854e-02, -1.2165150e-02, -2.0818561e-03],
```

Fig. 8 Model evaluation with KDD+aggregate

4.4. Construction of the IA-AE-IDS Model

The flowchart in Figure 9 presents the approach undertaken.

4.4.1. The IA-AE-IDS Dataset

To build our model, we compiled the dataset by capturing the traffic of an android mobile phone, Techno brand and Spark4 model over a period of one week. After processing, we retained this IA-AE-IDS dataset, the characteristics of which are presented in Table 7. In [1], we can see another step: digitalization, normalization and selection of attributes in detail.

Table 7. Summary of the IA-AE-IDS dataset

Dataset	Size	Attributes	Files	Size
IA-AE-IDS	8193	16	.csv	1020 Ko

4.4.2. Model Training and Export

The set of data being considered normal, the division of it allows us to obtain, on the one hand, the training data and the other hand, the validation data. Test data are missing. Thus, 80% is taken for training and 20% for validation.

Exporting the model consists of saving the different weights of the neural network in a file. This will make it possible to carry out the classifications in the following. We, therefore, proceeded to save the model, as shown in Figure 10.

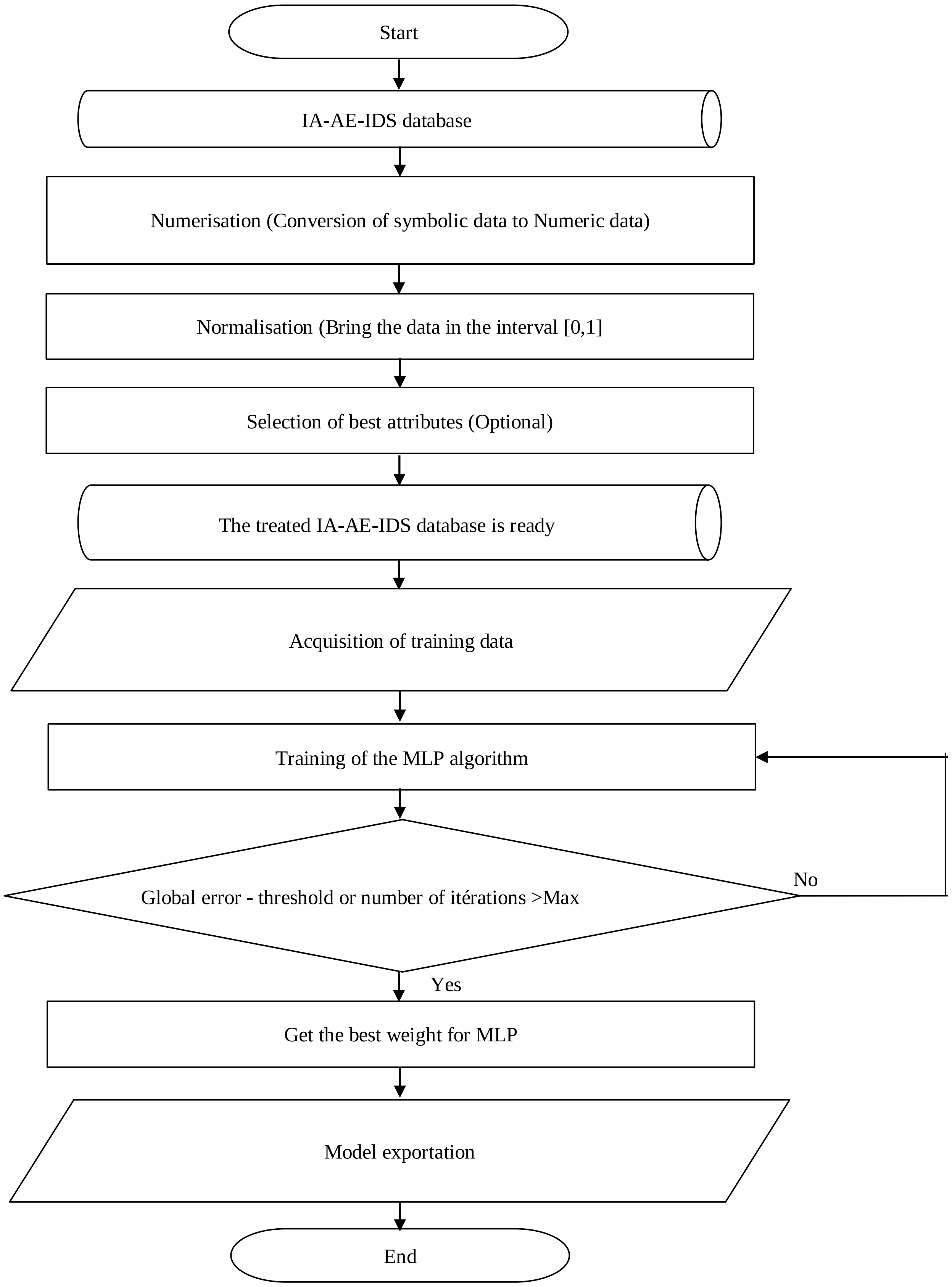
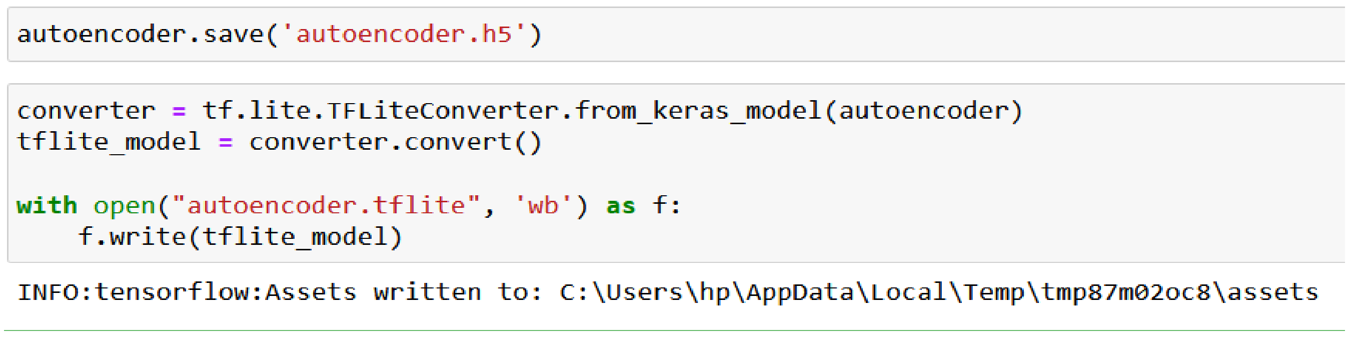

Start
IA-AE-IDS database
Numerisation (Conversion of symbolic data to Numeric data)
Normalisation (Bring the data in the interval [0,1]
Selection of best attributes (Optional)
The treated IA-AE-IDS database is ready
Acquisition of training data
Training of the MLP algorithm
Global error - threshold or number of itérations >Max
No
Yes
Get the best weight for MLP
Model exportation
End

Fig. 9 Model construction flowchart



```
autoencoder.save('autoencoder.h5')
```

```
converter = tf.lite.TFLiteConverter.from_keras_model(autoencoder)
tflite_model = converter.convert()

with open("autoencoder.tflite", 'wb') as f:
    f.write(tflite_model)
```

```
INFO:tensorflow:Assets written to: C:\Users\hp\AppData\Local\Temp\tmp87m02oc8\assets
```

Fig. 10 Exporting the IA-AE-IDS model

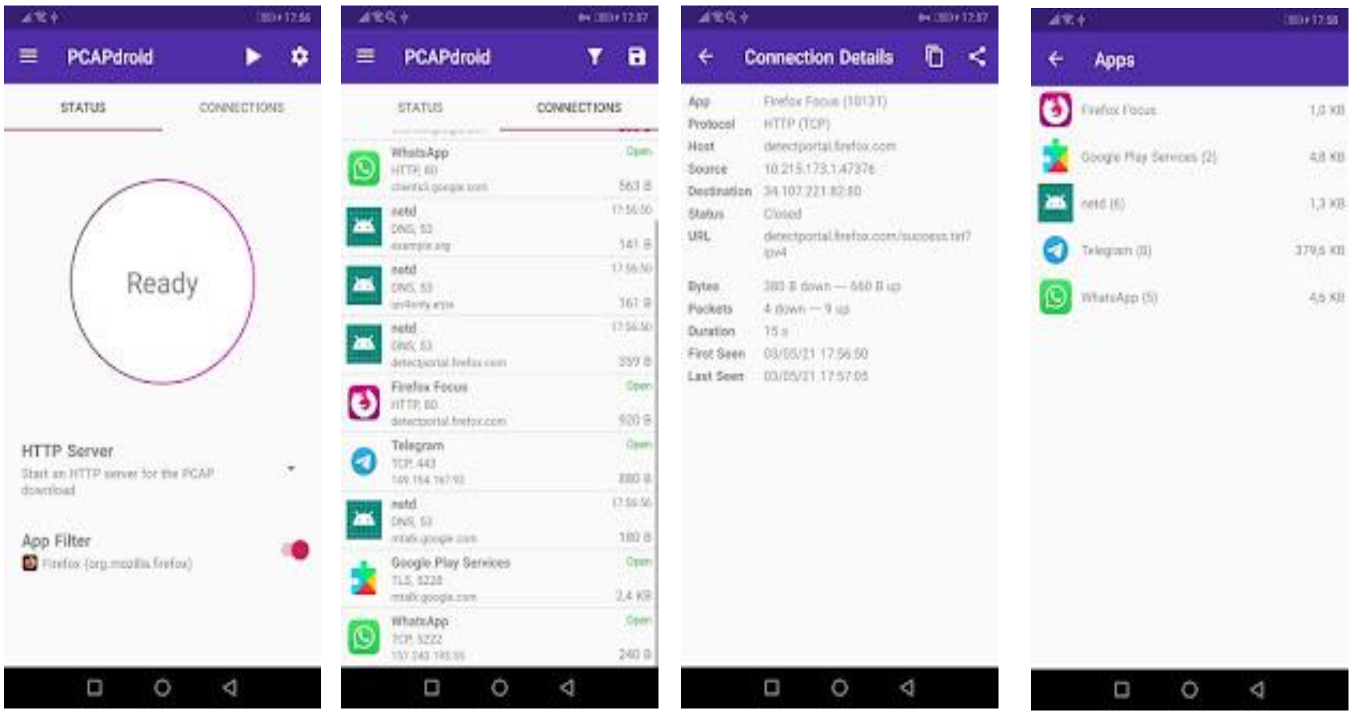
4.5. Model Deployment

To deploy the learned model, we chose a host android application to host our solution. This part will therefore consist of briefly presenting PCAPdroid and then integrating our solution.

4.5.1. Introducing PCAPdroid

As presented above, PCAPdroid is an open-source traffic analysis application shown in Figure 11. It offers several services [8]:

- Logs and examines user and system logins;
- Extract queries, URLs and remote addresses;
- Inspects http requests;
- Inspects connection payloads;
- Saves traffic to a PCAP file and/or sends it to a browser, stream, or remote analyzer;
- Offers an API to redirect traffic into another application.

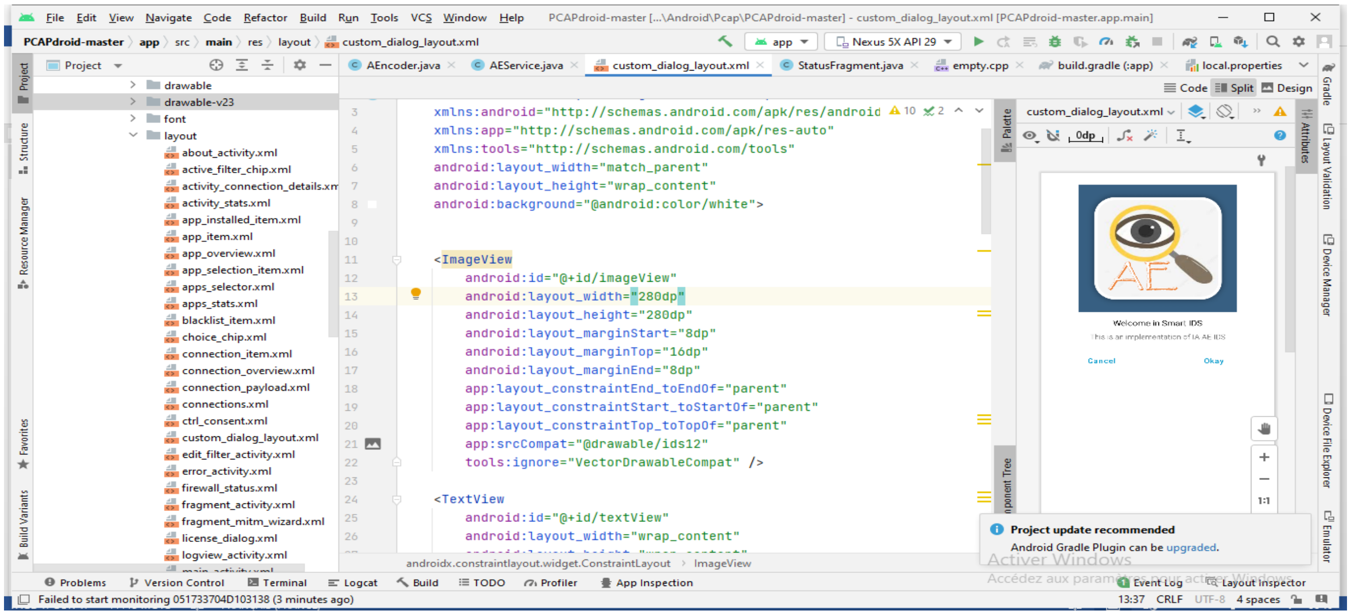


App Firefox Focus (10131)
Protocol HTTP (TCP)
Host detectportal.firefox.com
Source 10.215.173.1:47376
Destination 34.107.221.82:80
Status Closed
URL detectportal.firefox.com/success.txt?ipv4
Bytes 380 B down — 660 B up
Packets 4 down — 9 up
Duration 15 s
First Seen 03/05/21 17:56:50
Last Seen 03/05/21 17:57:05

Fig. 11 PCAPdroid

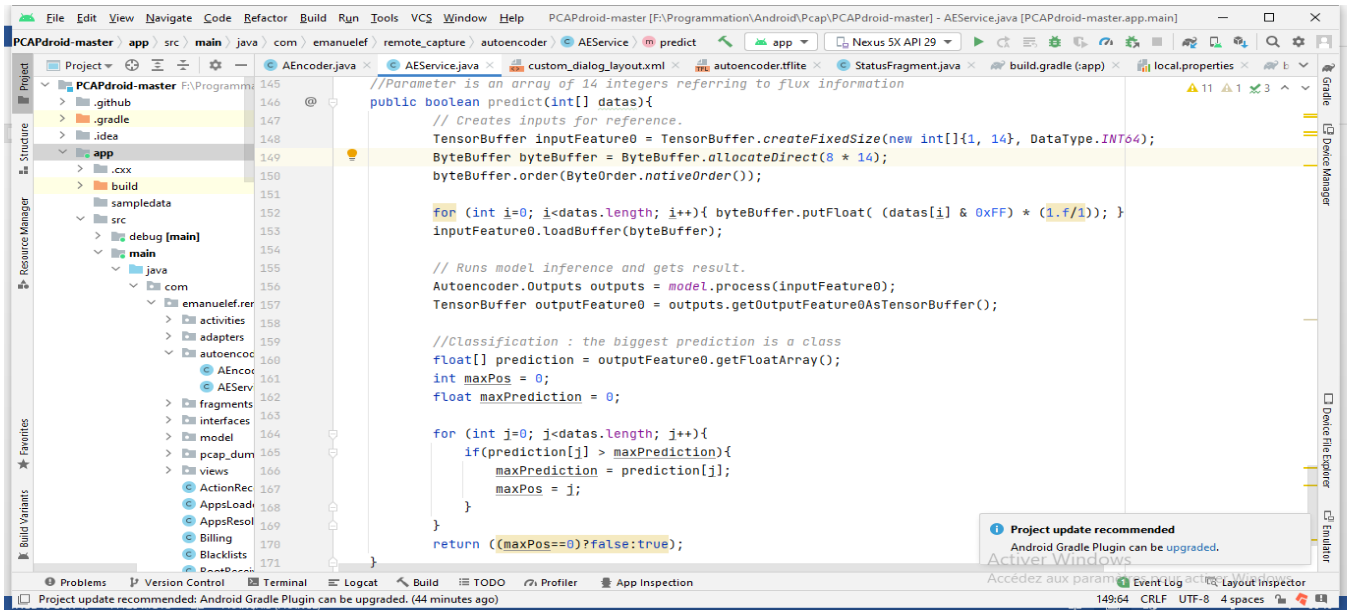
4.5.2. Construction of the Artificial Intelligence Layer

For the construction of this layer, we carried out work on two fronts.



```xml
    xmlns:android="http://schemas.android.com/apk/res/android
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="280dp"
        android:layout_height="280dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ids12"
        tools:ignore="VectorDrawableCompat" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
```

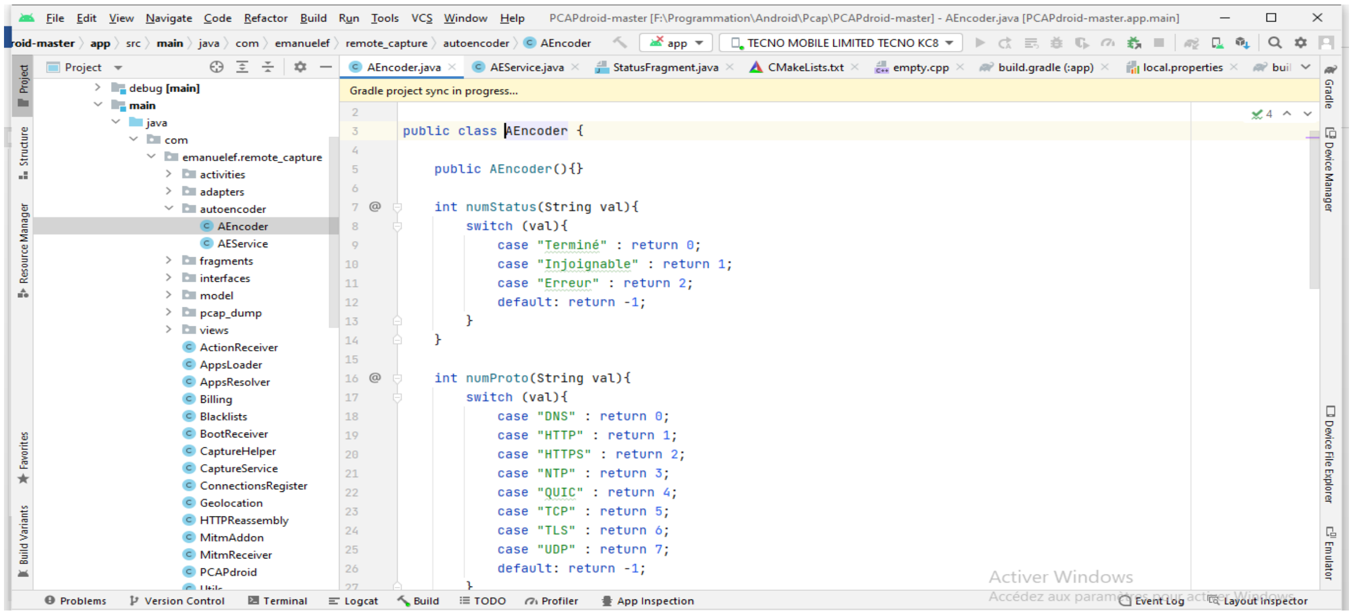
Fig. 12 Front-end dialog box



```java
    //Parameter is an array of 14 integers referring to flux information
    public boolean predict(int[] datas){
        // Creates inputs for reference.
        TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 14}, DataType.INT64);
        ByteBuffer byteBuffer = ByteBuffer.allocateDirect(8 * 14);
        byteBuffer.order(ByteOrder.nativeOrder());

        for (int i=0; i<datas.length; i++){ byteBuffer.putFloat( (datas[i] & 0xFF) * (1.f/1)); }
        inputFeature0.loadBuffer(byteBuffer);

        // Runs model inference and gets result.
        Autoencoder.Outputs outputs = model.process(inputFeature0);
        TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

        //Classification : the biggest prediction is a class
        float[] prediction = outputFeature0.getFloatArray();
        int maxPos = 0;
        float maxPrediction = 0;

        for (int j=0; j<datas.length; j++){
            if(prediction[j] > maxPrediction){
                maxPrediction = prediction[j];
                maxPos = j;
            }
        }
        return ((maxPos==0)?false:true);
    }
```

Fig. 13 Implementation of the back-end module

- Front-end : In the front end, we added an option to start and stop the IA-AE-IDS module to the home page interface. Indeed, considering the limited resources under Android, it was chosen to allow the user to control this module rather than leave it running throughout the application's life cycle, as shown in Figure 12. Also, we added some dialog boxes for displaying information.
- Back-end : For the back end, after importing the auto encoder, tflite model, we developed a service that would take care of receiving each instance of data, consulting its status and alerting the user through a dialog box in the case of the detection of an anomaly as shown in Figures 13, and 14.



Gradle project sync in progress...

```
public class AEncoder {

    public AEncoder(){}

    int numStatus(String val){
        switch (val){
            case "Terminé" : return 0;
            case "Injoignable" : return 1;
            case "Erreur" : return 2;
            default: return -1;
        }
    }

    int numProto(String val){
        switch (val){
            case "DNS" : return 0;
            case "HTTP" : return 1;
            case "HTTPS" : return 2;
            case "NTP" : return 3;
            case "QUIC" : return 4;
            case "TCP" : return 5;
            case "TLS" : return 6;
            case "UDP" : return 7;
            default: return -1;
        }
```

Fig. 14 Algorithm for analyzing traffic using the constructed model

5. Conclusion and Perspectives

At the end of this study in which, it was a question for us of providing a solution approach allowing smart phones to protect themselves against Spyware, with the contemporary giant Pegasus in the field of vision, it should be noted that a framework deploying an innovative model on an Android telephone, in order to analyze its traffic and signal possible presence of intrusion, has been proposed. Therefore, we first presented the study's context with the methods proposed in the literature, which allowed us to seek a problem-solving approach and its implementation.

We can remember that smartphones are currently an accessible target for attacks by hackers, particularly those aimed at confidentiality. From its architecture presenting a structure with critical resources, organizations, companies, and individuals consider and propose several solutions. Research in this direction is increasingly growing, and the proposals, in addition to resources, consider the type of data to be used and a preferred method. In this wake, we wanted to register by proposing the IA-AE-IDS detection system.

It is a system integrated into an Android application and consists of three modules. A probe that collects phone traffic for analysis. We produced an analyzer, an artificial intelligence module using an unsupervised algorithm called auto encoder, on a machine learning model called multilayer neural network and particularly on the multilayer perceptron network. A decision-maker, so the role is to alert the user when detecting intrusions.

As part of the outlook, we plan to improve the solution by testing the system to gauge its robustness, improve the approach and look for a better implementation without using the PCAPdroid host.

References

[1] P. Agarwal et al., "Malware Analysis in Mobile Devices," *7th International Conference on Computing in Engineering & Technology (ICCET 2022)*, India, pp. 128 – 134, 2022. [Cross Ref] [Google Scholar] [Publisher Link]

[2] Nataliya ZAGORODNA et al., "Network Attack Detection Using Machine Learning Methods," *Challenges to National Defence in Contemporary Geopolitical Situation*, vol. 2022, no. 1, pp. 55–61, 2022. [CrossRef] [Google Scholar] [Publisher Link].

[3] D.P. Gaikwad, Vismita Nagrale, and M.P. Bauskar, "Ensemble of Learner for Network Intrusion Detection System," *Journal of Network Security Computer Networks*, vol. 9, no. 1, 2023. [CrossRef] [Publisher Link]

[4] Emad Ul Haq Qazi, Muhammad Hamza Faheem, and Tanveer Zia, "HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System," *Applied Sciences*, vol. 13, no. 8, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] A Kalaivani, and R Pugazendi, "A Review on Intrusion Detection System and Its Techniques," *Data Analytics and Artificial Intelligence*, vol. 3, no. 2, pp. 132-137, 2023. [CrossRef] [Publisher Link]

[6] Maroua Ben Attia et al., "On-Device Anomaly Detection for Resource-Limited Systems," *30th Annual ACM Symposium on Applied Computing*, pp. 548-554, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[7] Frédéric Majorczyk, "*Détection D'intrusions Comportementales Par Diversification De COTS : Application Au Cas Des Serveurs Web Informatique*, " HAL Theses, Université Rennes, 2008. [Google Scholar] [Publisher Link]

[8] Maurras Ulbricht Togbe et al., "Etude Comparative Des Méthodes De Détection D'anomalies," *Revue Des Nouvelles Technologies De l'Information*, 2020. [Google Scholar] [Publisher Link]

[9] Safana Abbas, Wedad Abdul Khuder Naser, and Amal Abbas Kadhim, "Subject Review: Intrusion Detection System (IDS) and Intrusion Prevention System (IPS)," *Global Journal of Engineering and Technology Advances*, vol. 14, no. 2, pp. 155-158, 2023. [CrossRef] [Publisher Link]

[10] Hongmin Wang, Qiang Wei, and Yaobin Xie, "A Novel Method for Network Intrusion Detection," *Scientific Programming*, vol. 2022, pp. 1-13, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Kunpeng Wang, Jingmei Li, and Weifei Wu, "Intrusion Detection Algorithm Based on Transfer Extreme Learning Machine," *Intelligent Data Analysis*, vol. 27, no. 2, pp. 463-482, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Sihong Lin et al., "An Intrusion Detection Method Based on Granular Autoencoders," *Journal of Intelligent and Fuzzy Systems*, 2023. [Google Scholar] [Publisher Link]

[13] Anita Shiravani et al., "Network Intrusion Detection Using Data Dimensions Reduction Techniques," *Journal of Big Data*, vol. 10, no. 27, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Hong Qian et al., "A Novel Cyber Intrusion Detection Model Based on Improved Hybrid Sampling," *Transactions of the Institute of Measurement and Control*, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] T.P Deshmukh et al., "Smart Intrusion Detection in Industrial Devices Using Deep Belief Network," *International Journal of Scientific Research in Engineering and Management*, vol. 5, no. 4, pp. 2318-2323, 2023. [CrossRef] [Publisher Link]

[16] Nihar Mudigonda, "A Method for Network Intrusion Detection Using Deep Learning," *Journal of Student Research*, vol. 11, no. 3, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Reza Ghanbarzadeh, Ali Hosseinalipour, and Ali Ghaffaria, "Novel Network Intrusion Detection Method Based on Metaheuristic Optimization Algorithms," *Journal of Ambient Intelligence and Humanized Computing*, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Ali Muhammad et al., "Cortex-Inspired Ensemble Based Network Intrusion Detection System," *Neural Computing and Applications*, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Smrutirekha Panda, "Cyber Intrusion Detection, Prevention, and Future IT Strategy," *International Research Journal of Engineering and Technology*, vol. 9, 2022. [Google Scholar] [Publisher Link]

[20] Ngamba Thockchom, Moirangthem Marjit Singh, and Utpal Nandi, "A Novel Ensemble Learning-Based Model for Network Intrusion Detection," *Complex & Intelligent Systems*, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] Hyojoon Han, Hyukho Kim, and Yangwoo Kim, "Correlation between Deep Neural Network Hidden Layer and Intrusion Detection Performance in IoT Intrusion Detection System," *Symmetry*, vol. 14, no. 10, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] Taehoon Kim, and Wooguil Pak, "Deep Learning-Based Network Intrusion Detection Using Multiple Image Transformers," *Applied Sciences*, vol. 13, no. 5, pp. 1-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Alaa Mohammed Banaamah, and Iftikhar Ahmad, "Intrusion Detection in IoT Using Deep Learning," *Sensors*, vol. 22, no. 21, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] D. Kamalakkannan et al., "A Detection of Intrusions Based on Deep Learning," *Cybernetics and Systems*, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[25] Nisha T.N. Dhanya Pramod, "Insider Intrusion Detection Techniques: A State-of-the-Art Review," *Journal of Computer Information Systems*, 2023. [CrossRef] [Google Scholar] [Publisher Link]