ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ:
ΕΥΦΥΗ ΠΕΡΙΒΑΛΛΟΝΤΑ ΣΕ ΔΙΚΤΥΑ ΝΕΑΣ ΓΕΝΙΑΣ

BOT GENERATION AND DETECTION FOR SENSOR TIME SERIES DATA USING LSTM

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ανδριάνη Αθανάσιου

Επιβλέπων : ΚΩΣΤΟΥΛΑΣ ΘΕΟΔΩΡΟΣ

Μέλη εξεταστικής επιτροπής: ΒΛΑΧΟΥ ΑΚΡΙΒΗ
ΣΤΕΡΓΙΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ

Σάμος, Ιούνιος 2022

Πρόλογος και ευχαριστίες

Η παρούσα Διπλωματική Εργασία εκπονήθηκε, στα πλαίσια του Μεταπτυχιακού Προγράμματος Σπουδών " Διαδίκτυο των Πραγμάτων: Ευφυή περιβάλλοντα σε δίκτυα νέας γενιάς" του Πανεπιστημίου Αιγαίου.

Η εργασία πραγματοποιήθηκε υπό την επίβλεψη του κ. Θεόδωρου Κωστούλα, Αναπληρωτή Καθηγητή του Τμήματος Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων, του Πανεπιστημίου Αιγαίου.

Στο σημείο αυτό, θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν στην εκπόνηση της διπλωματικής μου εργασίας και στην ολοκλήρωση του προγράμματος. Αρχικά θέλω να ευχαριστήσω τον Κ. Κωστούλα που ήταν δίπλα μου σε ότι χρειάστηκα, σε τεχνικό, συμβουλευτικό και ψυχολογικό επίπεδο. Επίσης, τον κ. Χρήστο Ηλίου, για την καθοδήγηση του και την υπομονή του, σχετικά με το τεχνικό μέρος της εργασίας. Χωρίς τη συμπαράσταση και συνεχή βοήθειά τους, η ολοκλήρωση αυτής της εργασίας δεν θα ήταν δυνατή.

Ευχαριστώ ανεξαιρέτως, όλους τους καθηγητές μου στο πρόγραμμα για τις πολύτιμες γνώσεις και κατευθύνσεις, που μου έδωσαν.

Θέλω επίσης να ευχαριστήσω, τους συνάδελφους που γνώρισα στο πρόγραμμα και πλέον, καλούς μου φίλους Δημήτρη Τ., Γιώργο Π., Κωνσταντίνα Κ. για τα άπειρα ξενύχτια καραντίνας, που περάσαμε μπροστά από το viber, προκειμένου να βοηθήσουμε και να βοηθηθούμε, ο ένας από τον άλλο. Αν και ήμασταν μακριά, έμεινε σχέση ζωής. Δεν θα το ξεχάσω ποτέ. Σας ευχαριστώ.

Τους φίλους μου Γιάννη Ν., Μαρία Μ., Ευγενία Κ. και Γιάννη Μ. για την συμπαράσταση, την βοήθεια και το ενδιαφέρον.

Την οικογένεια μου, τους γονείς μου Γιώργο και Άννα, τον αδερφό μου Μανώλη και τη γιαγιά Βασιλεία, που σε όλη μου τη ζωή, μου παρείχαν τα πάντα, με κάθε τρόπο, για να φτάσω ως εδώ.

Την γυναίκα μου Μαρία για την συμπαράσταση, την βοήθεια σε ότι μπορούσε, την κατανόηση, την υποστήριξη και την υπομονή της.

Τέλος, όλη η εργασία, και η ολοκλήρωση του προγράμματος είναι αφιερωμένη στους δυο μου γιους, τον Γιώργο και τον Βασίλη.

Πίνακας περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

Ακρωνύμια

IoT	Internet of Things
GAN	Generative Adversarial Networks
LSTM	Long Short - Term Memory
BLSTM	Bidirectional Long Short-Term Memory
NN	Neural Networks meaning Artificial Neural Networks
ANN	Artificial Neural Networks
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
HuMIdb	Human Mobile Interaction database
GDPR	General Data Protection Regulation
AI	Artificial Intelligence
API	Application Programming Interface
BSD	Berkeley Software Distribution
GPU	Graphics Processing Unit
ML	Machine Learning
ReLU	Rectified Linear Unit
DL	Deep Learning

Περίληψη

Στην εποχή του IoT, οι συσκευές παράγουν τεράστιες και συνεχείς ροές πληροφοριών. Διερευνώντας τέτοιες ροές δεδομένων για νέα γεγονότα, προβλέποντας μελλοντικές εμπειρίες και αποφασίζοντας για δυνατότητες ελέγχου, χρησιμοποιούνται προγράμματα που αυτοματοποιούν την περιήγηση και εκτελούν ορισμένες εντολές που ονομάζονται bots. Σήμερα, τα bots απαιτούνται λόγω του τεράστιου διαθέσιμου περιεχομένου, αλλά έχουν χρησιμοποιηθεί και για διάφορους κακόβουλους σκοπούς, όπως επιθέσεις botnet, παραπληροφόρηση και χειραγώγηση διαδικτυακών συνομιλιών.

Τα bots που βασίζονται στην τεχνητή νοημοσύνη βρίσκονται σε άνοδο αυτές τις μέρες, με τη δυνατότητα να αναγνωρίζουν το μοτίβο χρήσης ενός χρήστη και να αναπτύσσονται ανάλογα. Ως αντίμετρο, στον τομέα της αποφυγής οποιασδήποτε μορφής επιθέσεων από bot, τα συστήματα που βασίζονται στη μάθηση είναι αναγκαία.

Σε αυτή τη μεταπτυχιακή διατριβή υιοθετείται, μια τεχνική για τη δημιουργία μιας χρονοσειράς αισθητήρων από ένα bot που δημιουργείται με μεθόδους τεχνητής νοημοσύνης και στη συνέχεια πρόκειται να ταξινομηθεί εάν μια χρονοσειρά προήλθε από bot ή άνθρωπο. Η δημιουργία του bot προήλθε από GAN, με NN και LSTM και στη συνέχεια ανιχνεύθηκε από Bidirectional LSTM, διερευνώντας την ακρίβεια και τη διαφορά, από συνθετικές έναντι πραγματικών χρονοσειρών, σε διαφορετικά σενάρια εκπαίδευσης, με εξαιρετικά αποτελέσματα μερικές φορές έως και 100%. Στο τέλος εξηγείται γιατί η δημιουργία, δεν λειτούργησε τόσο καλά λόγω converge failure και τι πρέπει να ληφθεί υπόψη για τη μελλοντική εργασία, κατά τον συντονισμό ενός GAN, προκειμένου να αποφευχθούν τα ίδια λάθη.

Λέξεις Κλειδιά: Ανίχνευση Bot, Δημιουργία Bot, LSTM, GAN, HuMIdb, Ανθρώπινη συμπεριφορά

Abstract

In the era of IoT, devices produce enormous and continuous information streams. Investigating such amount of data for new facts, forecasting future experiences, and deciding on control possibilities, programs are used that automate browsing and perform certain commands which are called bots. Nowadays, bots are required because of the vast amount of available content, but also have been used for malicious purposes, such as botnet attacks, misinformation and manipulation of online conversations.

Botnets based on artificial intelligence are on the rise these days, with the ability to recognize a user's behavioral pattern and deploy themselves as humans. As a counter measure, in the realm of botnet attack avoidance, learning-based systems are an unavoidable necessity.

In this master thesis there is an approach to generate a sensor timeseries from a bot which is generated by AI methods, and then is about to classify whether a time series, came from a bot or a human. The generation of the bot came from GANs with NN and LSTM, and then detected by Bidirectional LSTM by investigating the accuracy from generated vs real timeseries, in different training scenarios, with excellent results sometimes up to 100%. In the end explains why the generation did not work so well due to convergence failure and what must consider to the future work, during the tuning of a GAN, regarding the hyperparameters, in order to avoid the same mistakes.

Keywords: Bot Detection, Bot Generation, LSTM, GAN, HuMIdb, Humanlike behavior

1

Introduction

A large number of sensing devices collects and produces various physical information throughout time for a variety of fields and applications, nowadays that is called the Internet of Things (IoT) era. These devices will produce enormous and continuous information streams, very quickly, depending on the application's concept. Investigating such data streams for new facts, forecasting future experiences, and deciding on control possibilities is an important technique that makes IoT a positive commercial perspective and a personal satisfaction-enhancing innovation.

Due to these enormous amounts of data that are produced from numerous devices, there is the need of automate process and anticipation. So, there are programs that are installed, to automate procedures as data collection, preparation, anticipation which are called bots. Bots are required because of the vast amount of available content on the Internet. There are a lot of kind of bots, like webbots, which are programs that make routine, acts like browsing and perform certain commands on the side of the creator. Bots are often used for web indexing, website monitoring, commercial data extraction, and web information feed retrieval. Other types of bots are Social Media bots, download bots and ticketing bots. Bots perform these duties by repeatedly accessing Web servers for an extended period of time. On the other hand, giving in such bots, unrestricted access to Web servers, web content and services, is not considered as a decent concept.[41]

Bots despite that are a powerful tool for workers across the globe, also have been used for several malicious purposes, for example scraping material from webpages without permission, vulnerability scanning for malicious purposes, marketing and bank carding fraud, account takeovers, spamming, causing denial of service(DoS) attacks, and more. Bots can also be managed, most of times, without the owner's knowledge or agreement, from mobile phones and IoT devices, giving them a low-cost technique for scattered attacks. Advanced bots can potentially camouflage

their presence by impersonating human behavior. As a result, malicious bots have become a common hazard as well as a "good" bot is component of the Internet's infrastructure.[41]

Advanced bots are created in such a way that they cannot be detected by any common bot detection software. Therefore, as a countermeasure, sophisticated bots have been created to avoid detection.

In today's technologically advanced world, hacking into a remote system is no longer challenging. Malware has progressed to the point that it no longer requires writing to system files and may instead gain access to the system's mainframe via legitimate integrated apps. In terms of concept, botnets can be considered as fileless malware. Botnets do not require a malicious file to infect the target system. Their main objective is to disable or take control on, as many devices, which belongs to a network, as possible while obscuring the source. [39]

Because every item in the Internet of Things is connected to a local or bigger network, botnet attacks have grown to be a terrifying threat. Botnet attacks are considered as the next hacking weapon unless IoT companies upgrade their security procedures. Botnet assaults have become much more common in the last decade. Botnets are becoming more complex as technology progresses.

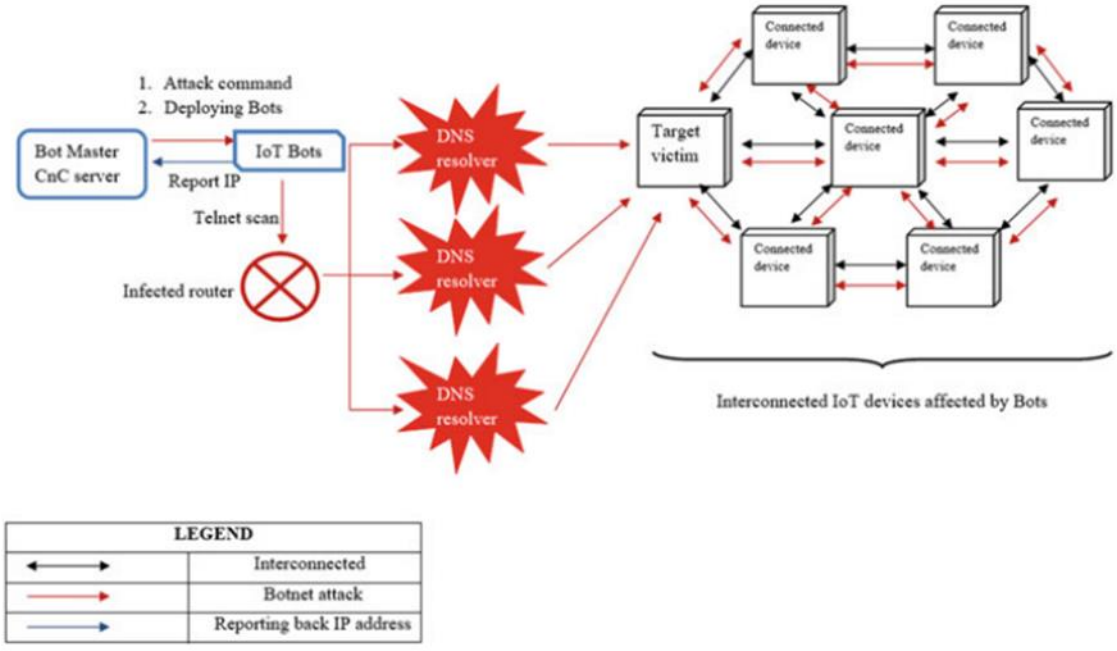


Figure 1:Existing problem of botnet attack on IoT network[66]

Many security vulnerabilities have arisen as a result of the broad usage of the Internet of Things. There have been malware assaults on IoT devices recently, such as the Torii botnet attack on September 2016, which was one of the most serious and intense of this kind of attacks, in order to take over a lot of IoT devices. But Mirai being the most famous because, in October 2016, Mirai malware attacked IoT devices such as IP cameras, DVRs, and routers, and large-scale DDoS attacks were launched utilizing those affected devices. The attack was directed at Dynamic DNS (Domain Name Service) servers and resulted in a throughput of around 1.2 Tbps. Amazon, Twitter, and Netflix[39] were among the major online services that were affected. On April 7th, 2017, at around

midnight, the city of Dallas was hacked. An attacker gained access to the Dallas city's emergency management system, triggering 156 loud alarms that were designed to alert the public in the event of a tornado[66].

DDoS attacks, phishing, and spamming may all be carried out using bots infected with Mirai or similar IoT malware. These assaults can cause long periods of network unavailability, resulting in financial losses for network businesses, as well as the leakage of personal data from users. In late 2016, Mirai infected over 2.5 million IoT devices, according to McAfee. Researchers calculated that at least 100,000 machines infected with Mirai or similar malware were disclosed everyday using telnet scanning telemetry data, according to Bitdefender's blog post from September 2017. Furthermore, many infected devices are projected to stay infected for an extended period of time. As a result, there's a strong incentive to identify these IoT bots and take necessary action against them so that they can't do any more harm. [39]

Bots, or automated social media accounts, have been used to propagate misinformation and manipulate online conversations. During the first impeachment of US President Donald Trump, looked at how bots behaved on Twitter. Despite bots accounting for only 1% of all Twitter users, bots were responsible for nearly 31% of all impeachment-related tweets, distribute more misinformation, and use less inflammatory language than other users. Bots were also found to be prevalent among proponents of the Qanon conspiracy theory, a prominent disinformation operation.[54]

Current botnet detection technologies, such as signature or flow-based error detection, performs well when a small-scale botnet attack occurs. On a larger scale, they are not achieving the expected results. That’s why, bots that are based on artificial intelligence are on the rise these days, with the ability to recognize a user's behavioral pattern and deploy - introduce themselves as humans. As a result, in the realm of bot detection, learning-based systems are an unavoidable necessity[41]

In this master thesis there is an approach in order to generate a sensor timeseries from a bot which is generated by AI methods and then is about to classify whether a time series, came from a bot or a human. The contribution of the study and the experiment is:

- How different schemas of GAN(with NN and LSTM) were generating gyroscope sensor timeseries as the proof of concept.
- Tries existing detection methods(BLSTM) by investigating the accuracy from generated vs real timeseries, in different scenarios of training and testing.
- Explains why this bot generation was not so accurate and what have to consider, during the tuning of a GAN, regarding the hyperparameters.

The rest of the paper is organized as follows. In Section 2 examines the previous work regarding, bot generation and detection. In Section 3 places the methodology, introduces the principles of analysis the methods and the technologies are used for the experiment. In Section 4 the experiment and the results. In Section 5 The discussion of the presented efforts and summarizing the findings of the experiments. Finally, Section 6 concludes the paper and outlining the opportunities for future work.

2

State of the Art

Bot detection has been studied extensively, with methods ranging from statistical machine learning to deep learning-based approaches. Social bots are now interactive and employ generative models to generate text. This section contains the most recent research.

2.1 Bot generation

Bots were first envisioned by Alan Turing in the 1950s[60] and have existed in many forms since the dawn of the Internet, ranging from partially managed algorithms like spam generators [25][54] to fully automated algorithms like chatbots, which are designed to hold a conversation with a human. Recent advances in natural language processing, ranging from simple neural networks to transfer learning approaches, have enabled bots to communicate with real people in such a way that it is impossible to tell whether a paragraph was created by a bot or a human [7].

Natural language generation is a prize in text generation research. There may be additional approaches in this area that are more intriguing than the ones now in use. The research continues by stating that GANs dominate the area of continuous data (images) [35].

GAN is one of the most prominent generative models that has made significant progress in generation tasks. GAN was first developed for continuous data, particularly image production [30], but it was quickly expanded to discrete and textual data [72]. Text generation is treated as a reinforcement learning process, with the state representing previously generated words, the action representing the next word to be formed, and the generator net representing a stochastic policy mapping current state to a distribution across the action space[72]. After the entire text has been generated, the text samples are sent into the discriminator network, a classifier that has been trained to differentiate generated text samples from realistic ones, in order to obtain reward signals for updating the generator network[30]. In a minimax game, the generator and discriminator try to maximize their own benefit while minimizing their adversary's gain. Other research focuses on one-dimensional time series, offering a new architecture called Time Series GAN (TSGAN) for modeling realistic time series data. The results show that TSGAN outperforms other methods[56].

2.2 Bot detection

Previous research has offered a variety of bot detection systems, with some attempting to uncover and apply more beneficial attributes in statistical machine learning algorithms. Other deep learning-based systems, on the other hand, can extract the most exact characteristics automatically throughout their training phase. In a more specific description, ML employs a collection of algorithms to read and learn data and make the best possible judgments based on it, whereas DL is a subset of ML that employs many layers to form an artificial neural network that can learn from data and make intelligent decisions.

2.2.1 Machine learning-based methods

By training on a set of datasets, these algorithms learn to identify bots based on a set of input features. As a result, the performance of these models is determined by the value of the features that are used.

Random forest [28,64], AdaBoost [2,9], K-nearest neighbor (KNN)[62], support vector machine (SVM)[23], and decision tree algorithm[36] are just a few of the statistical machines that can learn these attributes automatically as bot identification approaches.

In addition, reinforcement learning is used in an effective reinforcement learning-based detection system that is meant to detect and identify infected hosts in a P2P botnet, including new bots with formerly strange behavior and payload[8].

Many of these algorithms have been tested in various datasets of bots, including Twitter bots[37,42,46,48,64], data from online banking[68], and a gaming bot like "Yul-Hyul-Gang-Ho Online," a Korean MMORPG game. [19]. In the case of botnets, the data comes from well-known bots such as Mirai or peer-to-peer botnets[17,36,38] or HTTP log data obtained from lab's public Web server[34].

In the Internet of Things, there are datasets containing security flaws that were created by known vulnerabilities in 9 IoT devices, including security cameras, webcams, baby monitors, thermostats, and doorbells. There have been some outstanding research using simple machine learning algorithms like SVM[47], decision trees, and k-nearest neighbors (kNN)[11] that have shown excellent results.

2.2.2 Deep learning-based models

The most common neural networks are LSTM and CNN, which have shown promise in a variety of fields, including the bot detection challenge[5,14,21,42,51,73]. “Also, using a Bidirectional Long Short Term Memory based Recurrent Neural Network (BLSTM-RNN) on well-known botnet datasets like Mirai, udp, and dns[43],”as well as simpler methods like Deep Learning Neural Network multilayer perceptron with data that can be collected in a Software Defined Networking (SDN) environment, as well as simpler methods like Deep Learning Neural Network multilayer perceptron with data that can be[40].

Botnet detection models by deep reinforcement learning with a Deep Q-Network (DQN) were a really intriguing method with outstanding findings that can be deployed in practically every network that supports SDN[69].

There are publications in the field of IoT that use the encoding phase of a Long Short-Term Memory Autoencoder to minimize the feature dimensionality of large-scale IoT network traffic data (LAE). Deep BLSTM was used to analyze changes or anomalies in network traffic samples in order to appropriately identify them. To validate the efficacy of the suggested hybrid DL approach, extensive experiments are carried out on the BoT-IoT dataset. The deep BLSTM model is robust against model under-fitting and over-fitting despite the considerable reduction in feature size. In binary and multi-class classification scenarios, it also achieves strong generalization ability [41,52]. DL models for botnet identification can also be used at the ISP level to keep track of all Internet-connected IoT devices. Some of the techniques used at the ISP level are RNN, Identity Recurrent Neural Networks (IRNN), Bidirectional Recurrent Neural Networks (BRNN), LSTM, BLSTM, CNN, and CNN-RNN [66].

2.2.3 Hybrid models

When dealing with nonstationary time series data, traditional anomaly detection algorithms can detect shallow features, but they fail to detect outliers on deep features of huge time series data. As a result, hybrid methods are used to achieve the best potential results, such as using leveraged GANs as a semi-supervised task to improve bot detection via data augmentation [71].

One of these hybrid systems uses LSTMs to extract time-dimensional features from time series data, a GAN to extract deep features from normal data, and extreme gradient boosting (XGBOOST) to classify and export anomaly scores. The suggested approach has clear advantages in terms of extracting features and detecting anomalies in time series data [70].

A new framework called GANBOT is built in another paper, in which the generator and classifier are connected by an LSTM layer as a shared channel. The proposed framework outperforms existing contextual LSTM approaches by boosting bot detection probability, according to experimental results on a benchmark dataset of Twitter social bots [46].

Net-GAN has been introduced for sensor and network monitoring data. It's a network anomaly detection in time-series utilizing RNNs and GANs that discovers abnormalities in multivariate time-series by using RNNs to exploit temporal relationships. Net-GAN detects anomalies in complicated, difficult-to-model network monitoring data by discovering the underlying distribution of the baseline, multivariate data without making any assumptions about its nature [26,71].

2.2.4 HuMIdb

HuMIdb, a novel multimodal mobile database comprised of 14 mobile sensors collected from 600 users, was used to evaluate many studies.

Beginning with the use of HuMIdb, this section outlines the sensors usually found in modern smartphones and provides an overview of the various ways these sensors can be utilized to mimic human-smartphone interactions. Then, using a Siamese Neural Network architecture, they evaluated a biometric authentication system based on simple linear touch gestures, with highly encouraging results [4].

BeCAPTCHA, a CAPTCHA approach based on the analysis of touchscreen information received during a single drag and drop operation in combination with accelerometer data, was one

of the experiments with synthetic data from bots. BeCAPTCHA's purpose is to see if the drag-and-drop task was completed by a human or a machine. Fake samples were created using Generative Adversarial Neural Networks and handmade approaches to test the method. The findings show that mobile sensors could be used to describe human behavior and create a new generation of CAPTCHAs [5]. More experiments on HuMIdb are being conducted for passive authentication on mobile devices using behavioral biometrics and user authentication mechanisms [49,58] or user authentication in advanced web applications [34].

3

Methodology

3.1 Technologies

3.1.1 Python and Libraries

Python is a general-purpose programming language with a high level of abstraction. Guido van Rossum, the show's programmer, was reading the published scripts from "Monty Python's Flying Circus," a 1970s BBC comedy series. Van Rossum needed a name that was short, unusual, and a little mysterious, so he chose Python for the language he was developing at the same time.

Python is available for download for free, thus there are no sales data, and it can be found on a variety of websites and is included with numerous Operation Systems distributions. Since 1991, new, reliable releases have been released every 6 to 18 months.

"Python has continually been in the top10 of the most common and widespread, programming languages in the TIOBE Programming Community Index since 2003, and it is currently the most popular language, since October 2021. In 2007, 2010, 2018, and 2020, it was named Programming Language of the Year. Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Reddit and Spotify are mentioned like, just a few of the global companies, that using Python."

"Python is a programming language with a very basic and consistent syntax that is used by millions of people. All of this adds up to a spectacular and diverse ecology as well as documentation. Also, the language comes with a large standard library that covers multiple topics, like string processing, almost every internet protocol, including HTTP, FTP and SMTP, software engineering and operating system interfaces. Because there are so many third-party extensions and libraries available, it can be used to solve a wide range of problems. Python contains the most powerful, dynamic, and productive frameworks and libraries for a wide range of applications and fields" [63]

For the implementation of the experiment the following libraries were used:

- **NumPy** for multidimensional array objects[32].
- **Pandas** for data structures [44].
- **Matplotlib**[33] and **Seaborn**[45] for visualizations.
- **SciPy**[67] for a valuable set of, mathematical methods and functions.

- **Scikit-learn**, **Keras** and **TensorFlow** [1,18,50] for machine learning methods
- **TableEvaluator**[12] for evaluating AI models

3.1.2 Long Short - Term Memory (LSTM)

It's typical to employ a Recurrent Neural Network when working with sequential data (NLP, time series, etc). (RNN). RNNs are just a series of neural nets with the output of one serving as the input for the next.

RNNs are trained in the same way as any other neural network: node weights are updated proportionally to the product of all previous nodes' derivatives. As a result, if the derivatives of other nodes in the network are near to zero, their weight updates will be very tiny. The vanishing gradient problem is a phenomenon that hinders our model from learning.

Similarly, if our network's gradients are huge, the weight updates will be large as well. The exploding gradient problem is what it's called. This isn't normally an issue with basic neural networks. This happens to an RNN if it has several neural nets stacked in a sequential order.

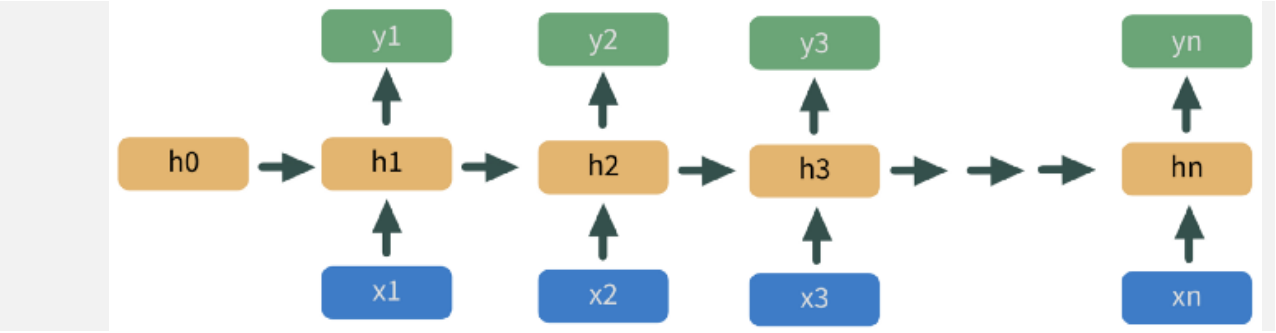


Figure 2:Diagram of a vanilla RNN [13]

Because RNNs contain more layers, the cell derivatives have more opportunities to compound. LSTM cells can help with this. LSTMs are a type of RNN cell that uses gates to govern incoming and outgoing data.

Gates are activation functions with a value ranging from 0 to 1. If a gate is given a low value, such as 0.001, multiply it by the incoming data and scale it down substantially. In the event that the gate takes on a large value, such as 0.998, multiply it by the incoming data and leave it mostly unscaled.

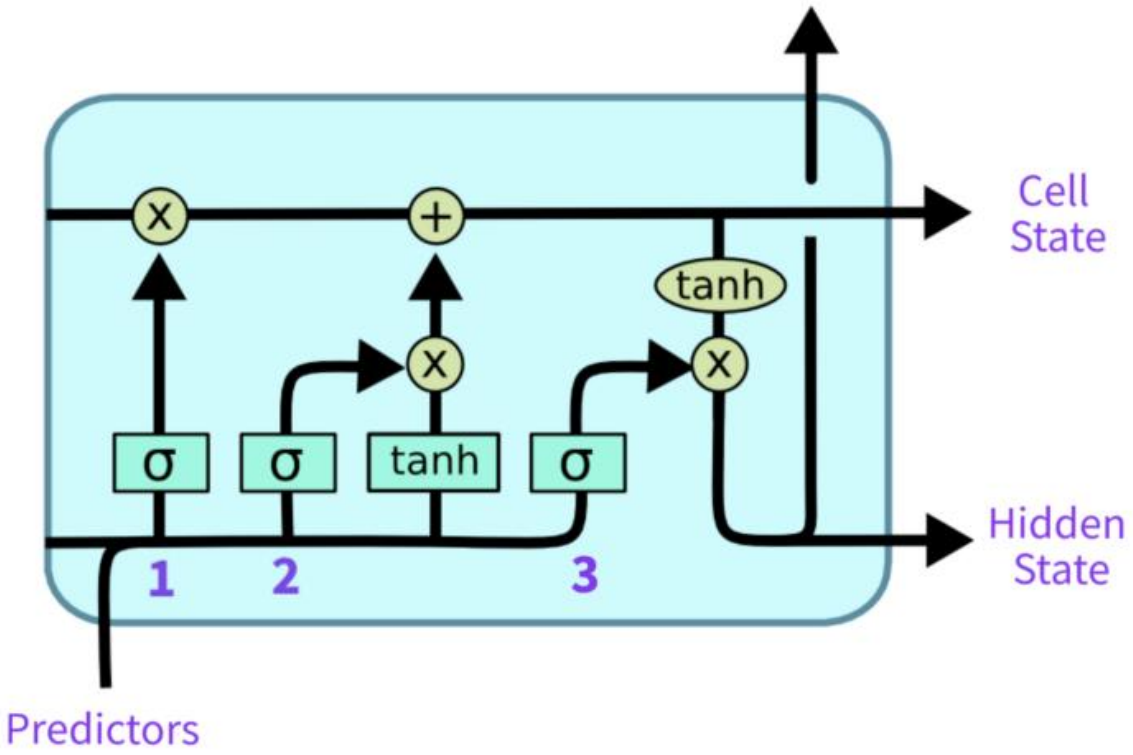


Figure 3:Components of an LSTM cell[13]

The teal boxes with a σ (sigma) in figure 3 are gates. They are neural networks that determine how much information from the previous cell and predictors should be used. It's worth noting that they're numbered 1–3.

The gates transfer this information to tanh after selecting how much information to maintain. Tanh functions are in charge of updating the node weights.

Finally, output two sets of data: the current state of the cell and the hidden state. The concealed state is the short term memory, whereas the cell state is the long term memory.

Let's quickly review the gates and define the numbers beneath each one.

1. Forget Gate: This gate determines whatever facts from the cellphone in that specific timestamp should be ignored. The sigmoid function is used to determine it.
2. Input Gate: determines the amount of this unit that is introduced into the current state. The sigmoid function decides which values to allow through 0,1, and the Tanh function assigns weightage to the values that are handed out according on their relevance level, which ranges from -1 to one.
3. Output Gate: determines which portion of the current cell is sent to the output. The Sigmoid characteristic determines which values to allow through zero, while the Tanh characteristic adds weightage to the values that can be exceeded by deciding their degree of importance, which ranges from -1 to at least one and is extended with the Sigmoid output. In other words, determine how much information should be output in our hidden state by combining long and short-term data.

By using these gates to regulate the flow of information, calculating problematic gradients can be avoided.

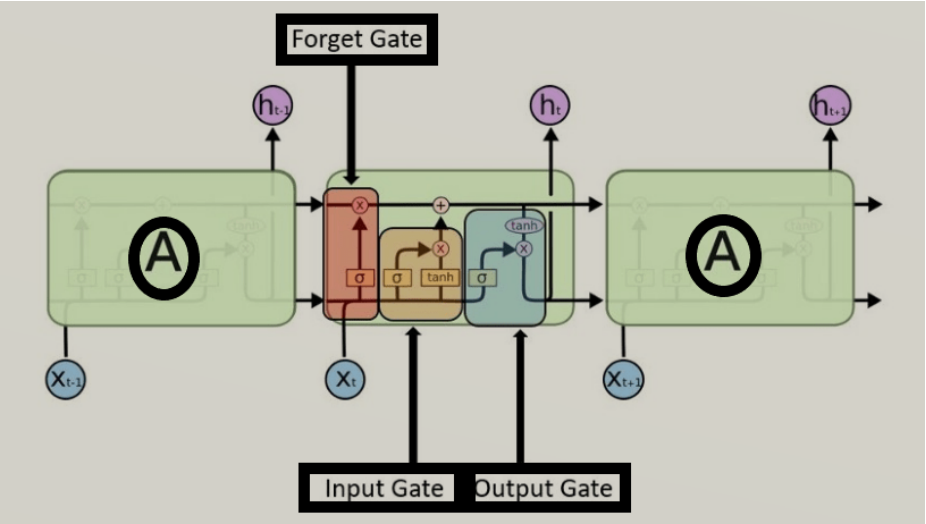


Figure 4:LSTM [59]

LSTM models are widely used in real-world applications and some of the most common are :

- **Text generation**. Is the process of computing language, from a given series of text. Such Language models can be used at multiple levels of grammar like character, single word, sentence and paragraph level.
- **Image processing**, which entails analyzing a photograph and converting the results into a text.
- **Speech Recognition**
- **Handwriting Recognition**
- **Time Series Forecasting**
- **Music generation.** Similarly, text generation, LSTMs anticipate musical notes as well, by studying a combination of provided notes, as input.
- **Language translation.** Is the process of converting a sequence from one language to another.

"Everything in our world has its own set of pros and cons, and LSTMs are no exception. LSTMs first gained popularity as a solution to the problem of disappearing gradients. However, it turns out that they were unable to totally remove it. The issue is that the data must still be moved from cell to cell in order to be evaluated. Furthermore, with the addition of other characteristics (such as forget gates), the cell has become fairly sophisticated."[27]

"They require a significant amount of time and resources to train and become suitable for real-world applications. In technical words, the system typically lacks the required amount of memory bandwidth due to the linear layers in each cell. As a result, LSTMs become inefficient in terms of hardware."[27]

Different random weight initialization affects LSTMs, making them act similarly to a feed-forward neural network. Instead, they prefer minimal weight initialization.

"Overfitting is a problem with LSTMs and using the dropout approach to solve it is complicated. During the training of a network, dropout is a regularization strategy in which input and recurrent connections to LSTM units are discarded from weight updates and hence from further training." [13,27,57,59].

3.1.2.1 Bi-directional long short term memory (BLSTM)

The act of making any neural network have sequence information in both ways backwards, meaning future to past, or forward, past to future, is known as bidirectional long-short term memory (BLSTM).

Our input runs in two directions in a bidirectional BLSTM, which distinguishes it from a standard LSTM. Input can only flow in one direction, either backwards or forwards, using a standard LSTM. However, with BLSTM, the input can flow in both directions, preserving both future and previous data.

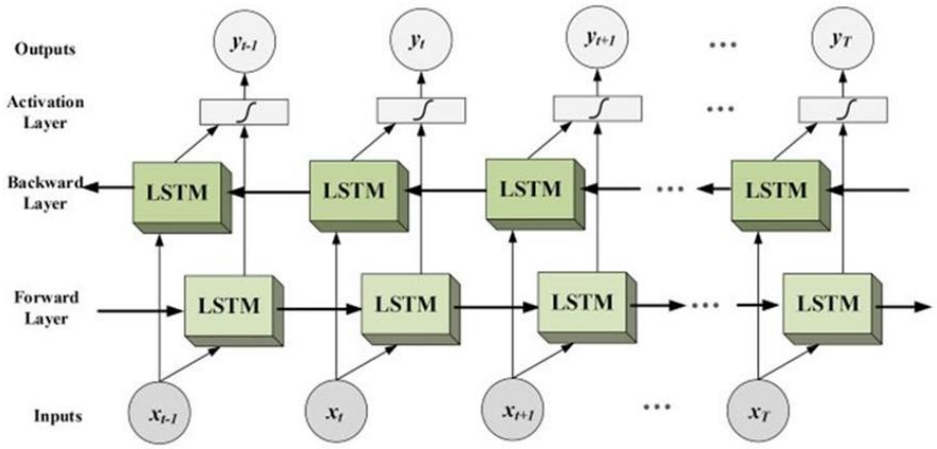


Figure 5:BLSTM[65]

3.1.3 Generative Adversarial Networks (GAN)

"GANs, or Generative Adversarial Networks, are generative models based on deep learning. GANs are a model architecture for training a generative model, and deep learning models are most commonly used in this." [30,31]

The GAN model architecture is made up of two sub-models:

- Generator. The model is used to produce fresh credible instances from the area of the problem.
- Discriminator. Model for determining if instances are genuine (from the domain) or not (generated).

"The generator network in generative adversarial networks competes against an adversary in a game theoretic setting. Samples are generated directly by the generator network. On the other

hand, discriminator tries to figure out the difference between, the real samples that are taken from the training dataset and fake samples, taken from the generator."[30,31]

3.1.3.1 The Generator Model

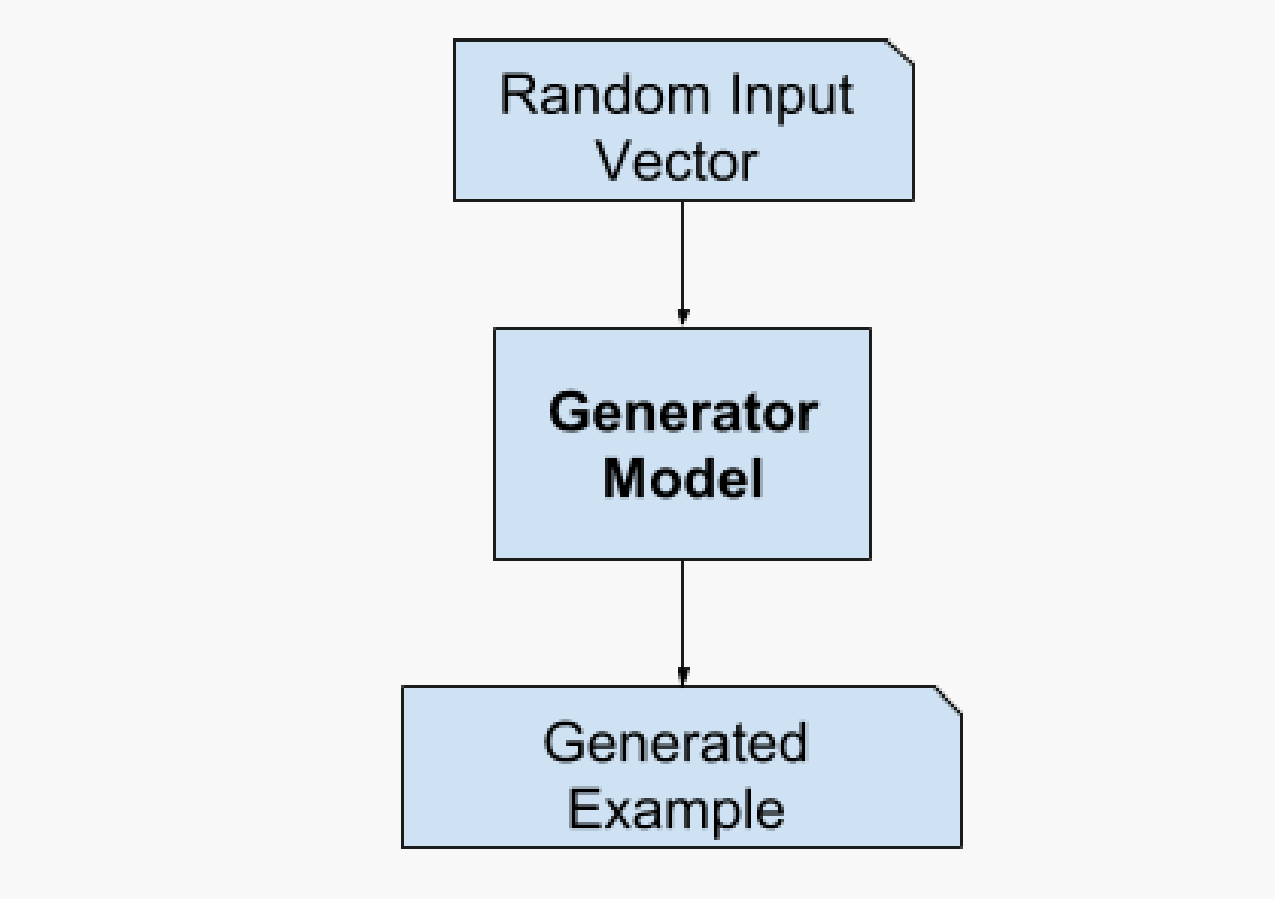


Figure 6:Example of the GAN Generator Model[30]

"The generator model creates a sample, in the domain using a fixed-length random vector, usually named, noise, as its input. A Gaussian distribution is employed to generate the vector, which is then used to start the generative process. In this multidimensional vector space, all the points will correspond to other points in the issue domain during training, resulting in a solid representation of the data distribution." [30,31]

"A latent space, or a vector space with latent variables, is the name given to this vector space. Latent variables, often known as hidden variables, are variables that are relevant to a domain but are not easily visible.Data distribution can be referred to as latent variables, or a latent space. A latent space, in other words, provides compression or high-level ideas of observed raw data, such as the input data distribution. In the case of GANs, the generator model assigns meaning to points in a given latent space, allowing new points to be drawn from the latent space as input and utilized to produce new and varied output examples." [30,31]

"Machine-learning models may learn the statistical latent space of images, music, and stories, and then sample from it to create new artworks with comparable qualities to those present

in the model's training data. After training, the generator model is kept and used to generate new samples.“[30,31]

3.1.3.2 The Discriminator Model

“The discriminator model takes a domain example, which can be actual or produced by the generator, as input and predicts whether it is authentic or fake.

The real-world example is taken from the training data set. The generator model produces the generated examples.

A conventional (and well-understood) classification model serves as the discriminator.

The discriminator model is usually removed after the training phase because the generator is the most interesting aspect of the GAN.

Because is trained to, extract characteristics from examples in the issue area, the generator can sometimes be repurposed. Using similar input data, some feature extraction layers can be used in transfer learning applications.”[30,31]

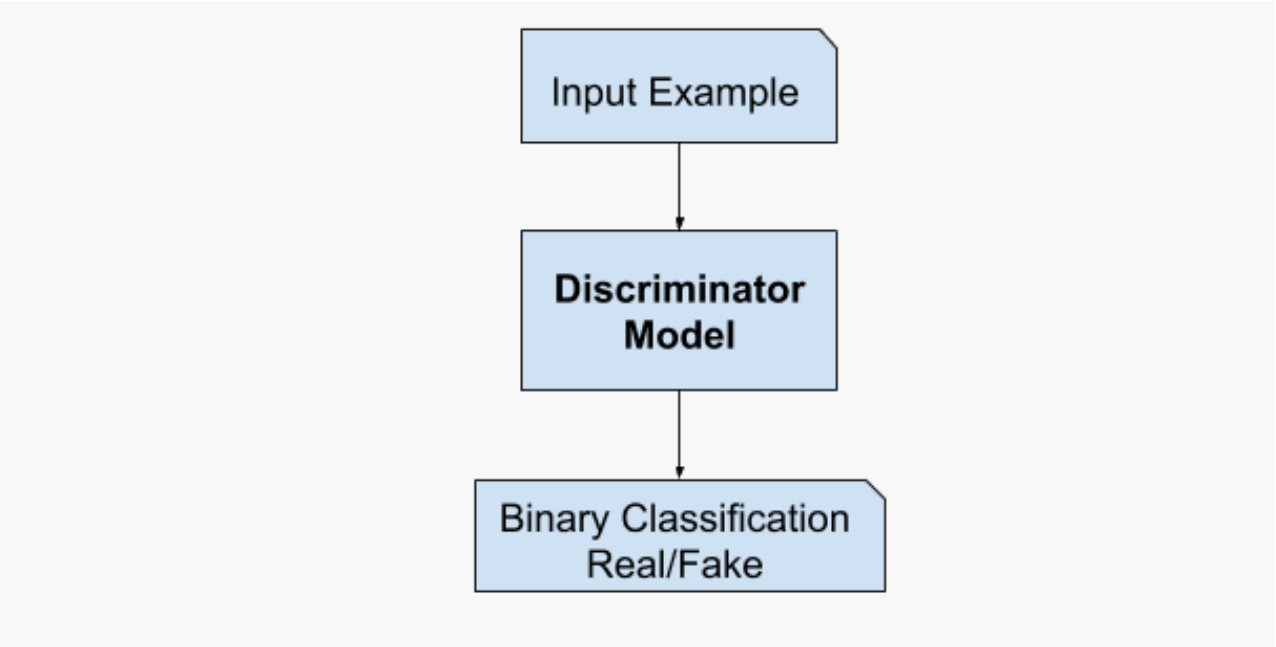


Figure 7:Example of the GAN Discriminator Model[30]

3.1.3.3 GANs as a Two Player Game

“Generative modeling is an unsupervised learning problem, but a brilliant characteristic of the GAN architecture is that the generative model's training is set up, as a supervised learning problem.

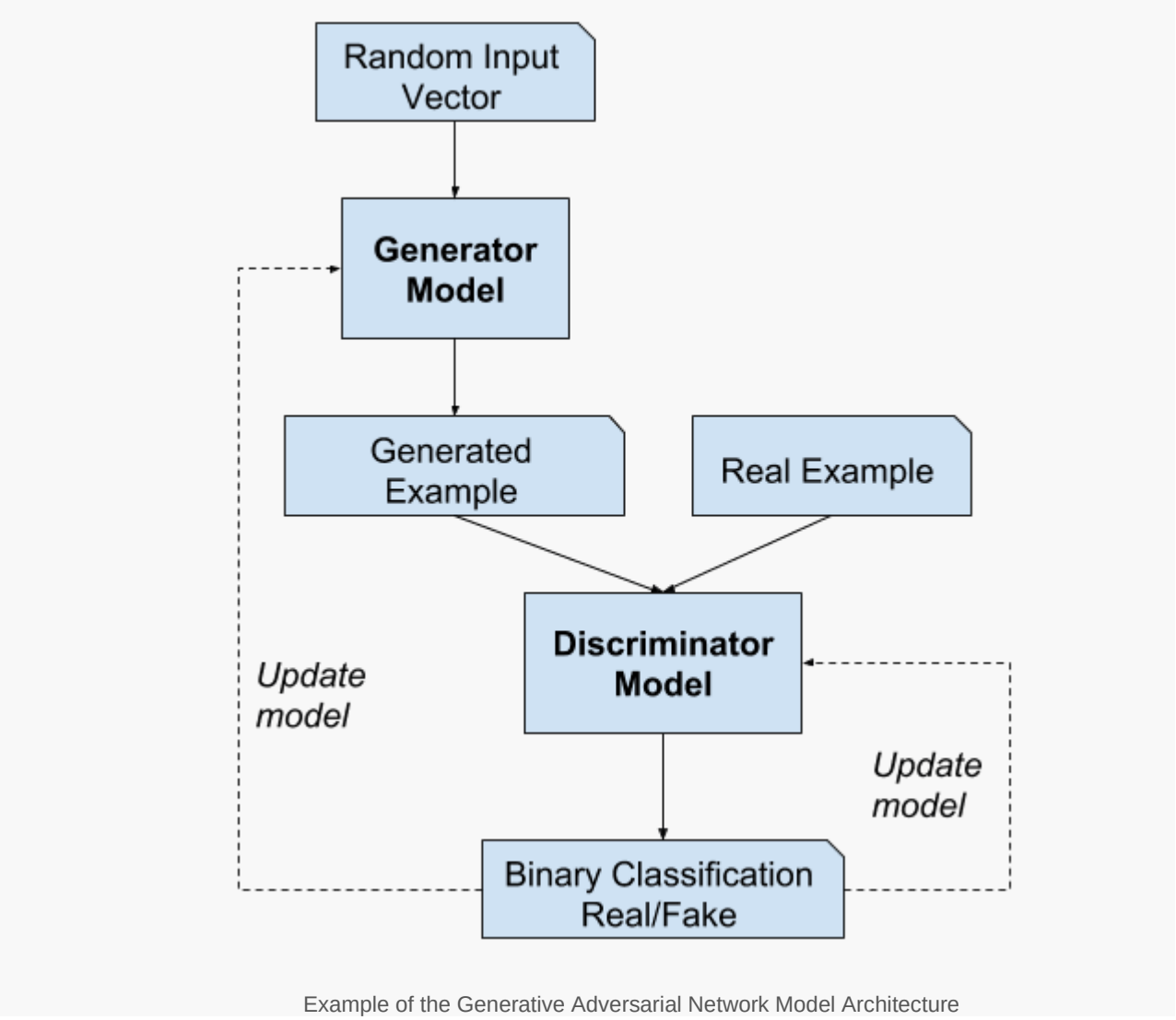
The generator and discriminator models are trained jointly. The generator creates a batch of samples, which are then given to the discriminator, alongside with real instances from the domain, to be classified, as true or false.

In the next iteration, the discriminator is changed to improve its ability to distinguish between real and false samples, while the generator is updated, based on how successfully, the

generated samples, deceived the discriminator. The two models are thus competing against one other, are antagonistic in the sense of game theory and are playing a zero-sum game. The GAN framework can be described as "adversarial" since it can be evaluated using game theory methods.

When the discriminator, successfully separates, between true and false samples, it is rewarded and there is no need of changes to the model parameters. Meanwhile the generator is penalized and model parameter updates. On the other hand, if the generator frauds the discriminator, it is rewarded, or the model parameters are not changed, au contraire the discriminator is penalized and its model parameters are updated.

At a certain point, the generator makes flawless copies from the input domain every time, and the discriminator can't detect the difference, thus it always predicts "unsure", meaning, 50% for real and fake.” [30,31]



Example of the Generative Adversarial Network Model Architecture

“The discriminator is taught to try to figure out how to tell whether a sample is genuine or not. At the same time, the generator is attempting to persuade the classifier that its samples are real. At convergence, the generator's samples are indistinguishable from real data, and the discriminator always returns 1/2. The discriminator can then be discarded.”[30]

3.2 Dataset

"HuMIdb is a novel multimodal mobile database that contains more than 5 GB from a a lot of mobile sensors acquired under unsupervised scenario". [3]

3.2.1 Description Of HuMIdb Database

"During realistic human-mobile contact, more than 600 users used 14 sensors (accelerometer, l. accelerometer, gyroscope, magnetometer, orientation, proximity, gravity, light, touchscreen, keystroke, GPS, WiFi, Bluetooth, Microphone). An Android application has been developed for the collection of the data. The data are sensor signals, which produced, while users do 8 easy activities, with their own smartphones and without any supervision (i.e., the users could be standing, sitting, walking, indoors, outdoors, at daytime or night, etc.). The acquisition app was released on Google Play and promoted through our research website and several research email groups. The participants were then self-selected from all around the world, resulting in a more diverse group of people than prior state-of-the-art mobile databases. According to the GDPR, all data collected in this database was retained on private servers and anonymized with prior participant approval."[3]

Sensors	Sampling Rate	Features	Power Consumption
Accelerometer	200 Hz	x, y, z	Low
L.Accelerometer	200 Hz	x, y, z	Low
Gyroscope	200 Hz	x, y, z	Low
Magnetometer	200 Hz	x, y, z	Low
Orientation	NA	l or p	Low
Proximity	NA	cm	Low
Gravity	NA	m/s^2	Low
Light	NA	lux	Low
TouchScreen	E	x, y, p	Medium
Keystroke	E	$key,\ p$	Medium
GPS	NA	Lat., Lon., Alt., Bearing, Accuracy	Medium
WiFi	NA	SSID, Level, Info, Channel, Frequency	High
Bluetooth	NA	SSID, MAC	Medium
Microphone	8 KHz	Audio	High

Figure 8:Description of all sensor signals captured in HuMIdb

"The sensors provides data from measure specific tasks that included in the HuMIdb app. The tasks are:

a. Keystroking: gets keystroke actions from fixed and free text
b. Swipe up: the users have to perform, swipe up gestures to complete tasks
c. Tap and double tap, is focused on tap gestures
d. Swipe down: the users have to perform, swipe down gestures to complete tasks
e. Circle hand gesture, is designed to draw in the air with the smartphone a circle
f. Cross hand gesture is designed to draw in the air with the smartphone a cross
g. Voice, records the user saying 'I am not a robot', and finally
h. Finger handwriting in which the user has to draw with the finger the digits 0 to 9 over the touchscreen

It's worth noting that all 14 sensors are obtained throughout the execution of all tasks, while some play a crucial part in a few of them. The accelerometer signal, for example, is recorded throughout the session, even if it may be more useful in activities e and f. For each task, this diverse data can be used to enhance the patterns, taken from the main sensor. In addition to the swipe patterns, all tasks include a right swipe button that is gained." [3]

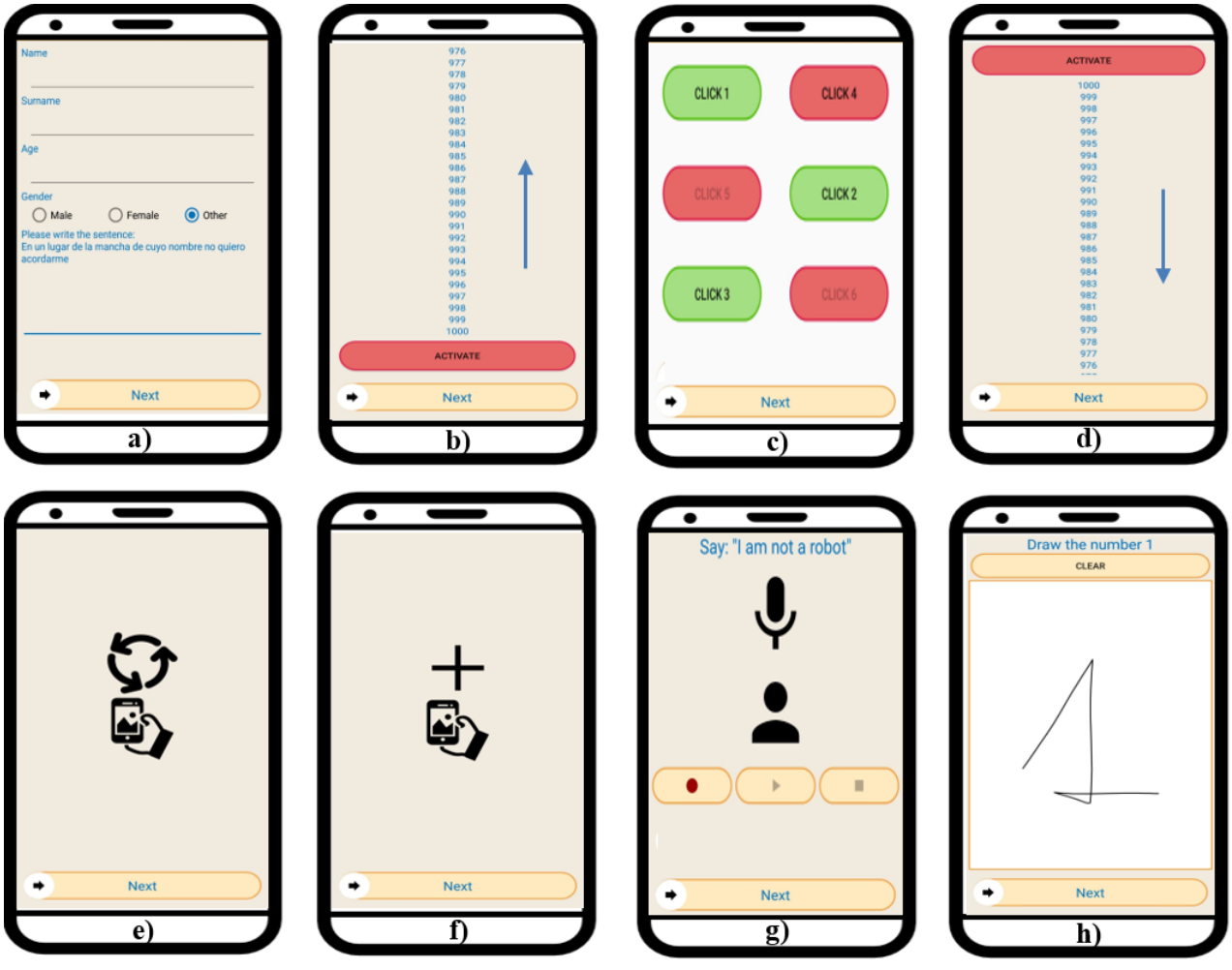


Figure 9:The mobile interfaces designed for the 8 mobile HuMI tasks

"The acquisition process consists of five sessions separated by at least one day. The app displays a quick pop-up message at the start of each job that explains how to perform it. The application also records the smartphone's orientation (landscape/portrait), screen size, resolution, model, and the date.

In terms of age, 25.6 percent of users were under the age of 20, 49.4 percent were between the ages of 20 and 30, 19.2 percent were between the ages of 30 and 50, and the remaining 5.8 percent were beyond the age of 50. In terms of gender, 66.5 percent of the participants were men, 32.8 percent were women, and 0.7 percent were unidentified. Participants used 179 different devices to complete tasks from 14 different nations (52.2 percent /47.0 percent /0.8 percent are European, American, and Asian, respectively).

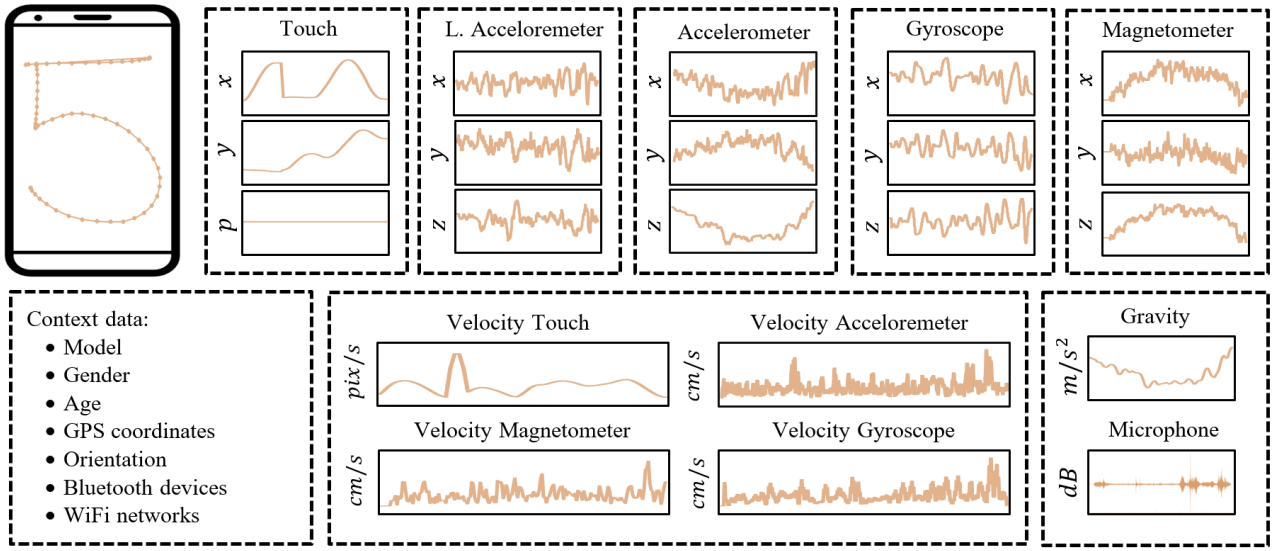


Figure 10:Full set of data generated during one of the HuMIdb task

Figure 10 shows an example of a handwriting task for the digit "5", as well as the data gathered during the task. It's interesting to see how a basic activity can generate a heterogeneous flow of data about the user's behavior, meaning the way the user holds the device, the power and velocity of the gesture, the location, and so on."[3]

3.2.2 HuMIdb Structure

"The data is saved in nested folders with the ID number used to identify each user's folder in HuMIdb. There are between 1 and 5 folders in the user's folder corresponding to the various sessions the user has completed, as well as 3 CSV files containing the Bluetooth, WiFi, and GPS data signals acquired during the entire session. Finally, for each activity, there is a folder in each session that holds all of the sensors collected during the task. The right swipe button is represented by the'swipe.csv' file."[3]

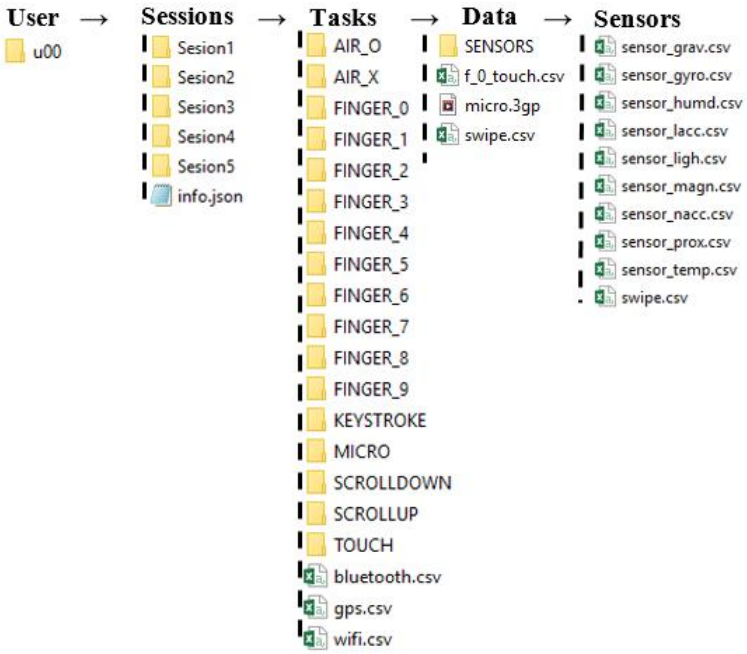


Figure 11:Structure of the nested folders of HuMi database[3]

"Each sensor's csv file contains data on the measurements taken by the sensor during a certain session. Sensor gyro.csv files, for example, have five columns that represent timestamp, orientation, and the gyroscope's x,y, and z axis data, respectively." [3]

4

Experiments

The main goal of the experiment is the creation of synthetic, time series data, that appears like they come from a sensor e.g., a gyroscope. This can be happened by using GANs and consider this as bots. LSTM and ANN, GANs were used to generate Sensor Data.

Then detect if a specific sensor-Data-Time-series, is from real or synthetic data, with the usage of a LSTM model, as classifier.

The Data is taken by HuMIDB Database (https://github.com/BiDAlab/HuMIdb). [3]

The source code is uploaded in GitHub (https://github.com/Anakinbarca/Bot-Detection)

4.1 Data Preparation

The first step is to visualize a part of the data, in order to provide a better view of the dataset. The visualization took care for each sensor, to choose which sensor and from which activity should use, in order to provide a challenging sequence to the GAN. In the next picture the 3 axis(x,y,z) of the gyroscope for each activity, are visualized.

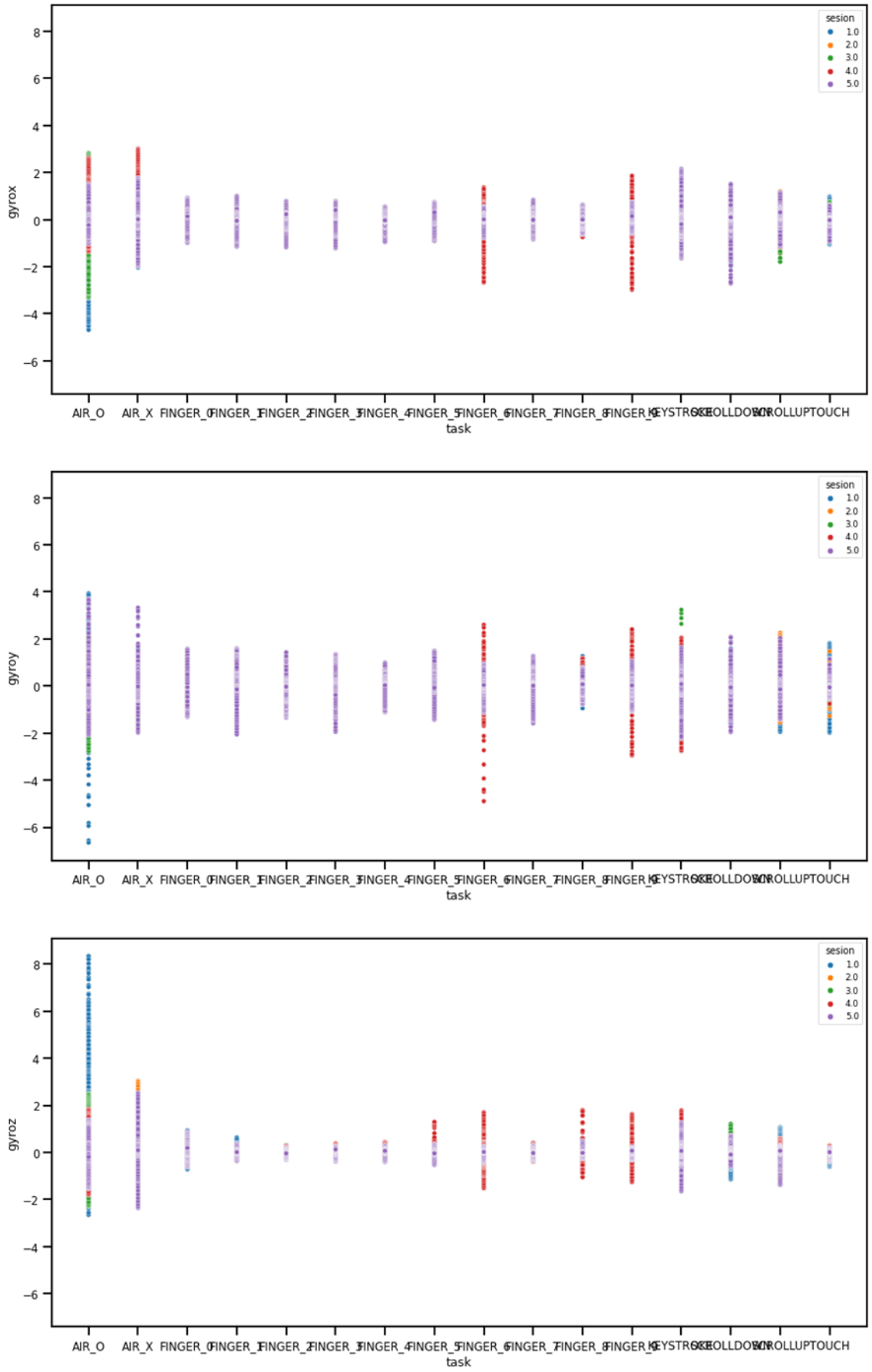


Figure 12:Gyroscope sessions visualization

The AIR_O activity seems like a good choice, due to the range of its values, in the different

sessions.

The next step is to merge the needed data from different persons in a csv file, from the different folders which are provided in the initial form of the dataset. This achieved from crawling files and folders, provided by HuMIDB, with python and csv file for importing data in DataFrames is easily produced.

The selected data, meaning the 3 axis of the gyroscope and their timestamp are resampled to 810 samples (average). Truncate and pad the input sequences, is needed so that they are all the same length for modeling. Resampling is used in time series data as a convenience method for frequency conversion. After their values are scaled at the range 0 to 1.

"The resampled signal begins at the same value, as the start of the sample, but the signal is presumed to be periodic because the Fourier method is applied. The argument window is a Fourier-domain window that shapes the Fourier spectrum before zero-padding to reduce ringing in resampled values for sampled signals that aren't meant to be understood as band-limited.

The signal.resample method from the scipy Python package is utilized for the resample. This function uses the Fourier method to resample a range of x to range of y samples along the provided axis. Upsampling and downsampling are terms used to describe when Y is higher or smaller than x." [67]

In upsampling the samples that are created are more than the given ones. In the following picture there is an example of upsampled data compared to real:

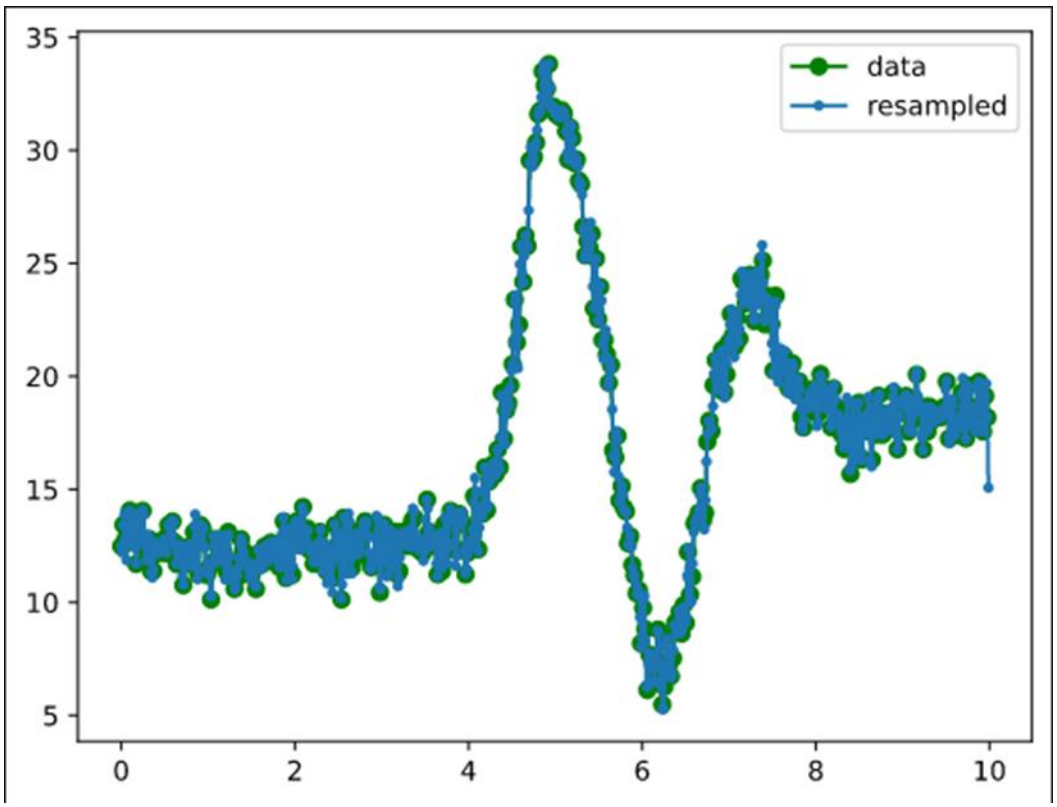


Figure 13:Upsampling

In downsampling the samples that are created are less than the given ones. In the following picture there is an example of downsampled data compared to real:

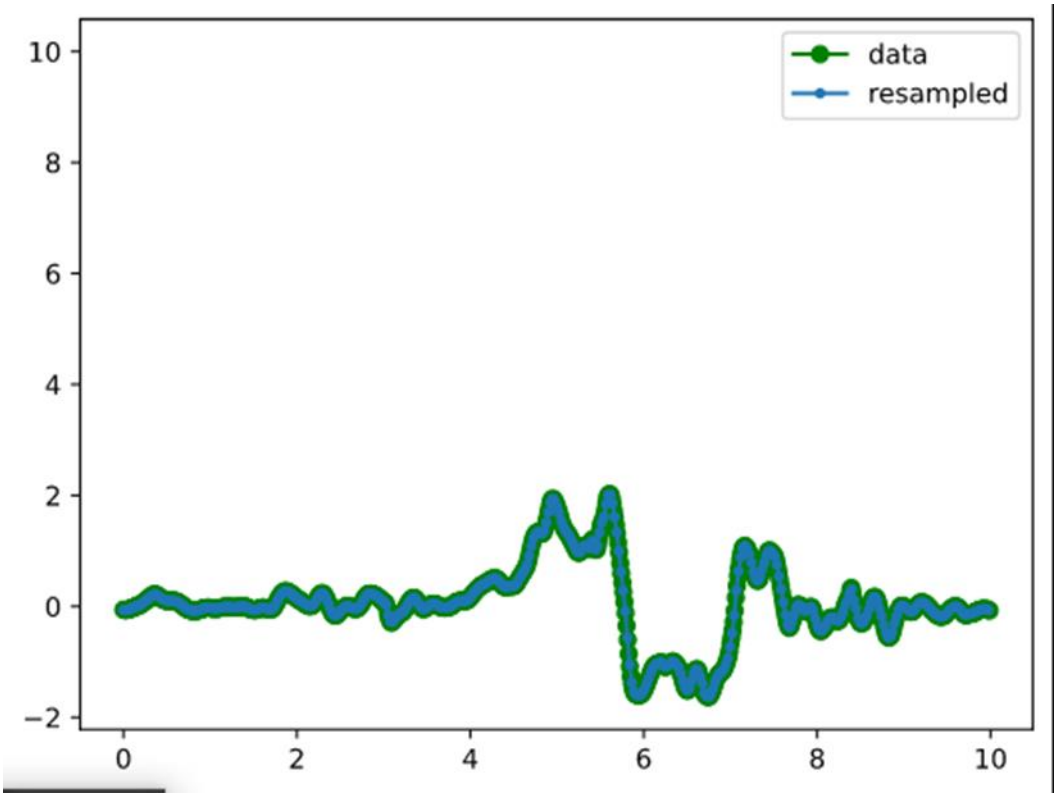


Figure 14:Downsampling

In both cases the library works perfectly.

4.2 Creation of synthetic data using GANs

With the taken data, two types of GANs are created in order to produce synthetic data for the 3 axis of the gyroscope and their timestamp (4 features total).

One GAN, by using Artificial Neural Networks and one by using LSTM. During the GAN training procedure, the models of generator and discriminator are stored separately, to get used later.

The generators were used, for synthetic samples creation (bots) and the discriminators as possible classifiers, even though is expected that they will not work correctly.

4.2.1 GAN creation with ANN

In the case of the simple GAN that both generator and discriminator are created only with ANNs, where used only dense layers. In any neural network, a dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. This is the most commonly used layer in artificial neural network networks.

The dense layer's neuron in a model receives output from every neuron of its preceding layer, where neurons of the dense layer perform matrix-vector multiplication. Matrix vector multiplication is a procedure where the row vector of the output from the preceding layers is equal to the column vector of the dense layer. The general rule of matrix-vector multiplication is that the row vector must have as many columns like the column vector.

The input is one of the time series (gyroscope sensor, of the 1st session, of the 1st person as is in the folders of the HuMIDB) separated, in batch of the size of 32 samples. The generator schema is represented in the picture below. The activation for every layer is ReLU.

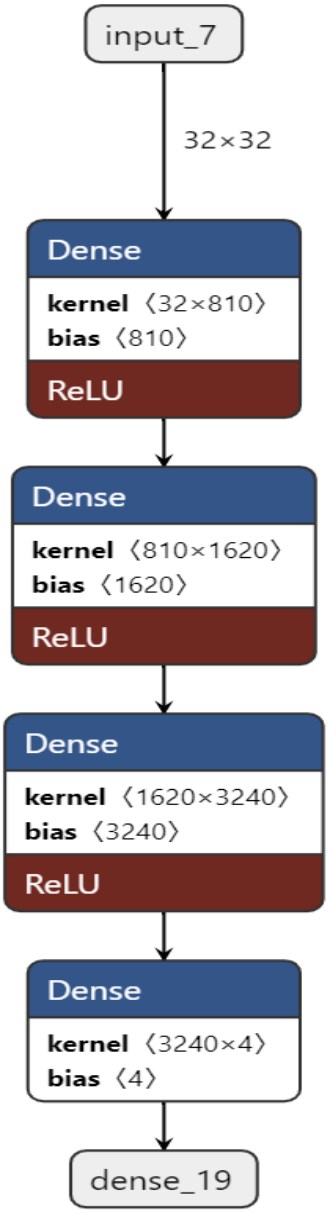


Figure 15:Neural Network GAN Generator Shape

In a similar way, the discriminator only in the output layer uses the sigmoid activation. The rest uses ReLU. Discriminator schema is represented in the picture below.

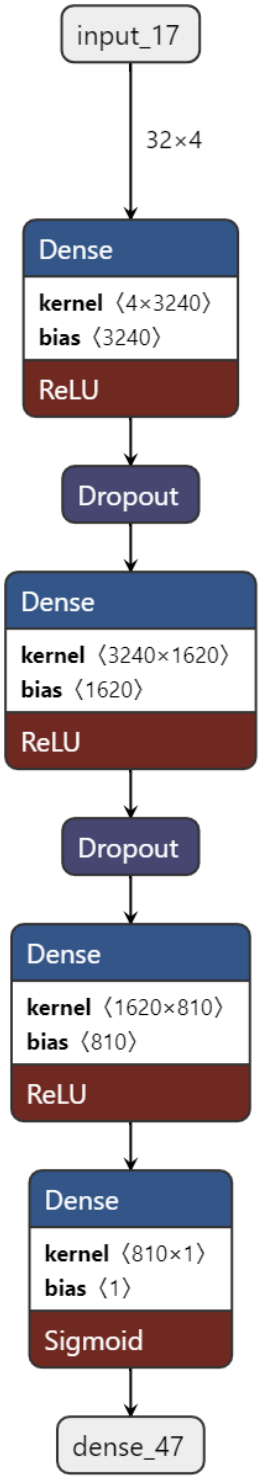


Figure 16:Neural Network GAN Discriminator Shape

Optimizer is Adam with beta at 0.5, loss is binary_crossentropy and the learning rate was 0.0005.

The results were not so encouraging as the Generator Loss was at the 1.30, while the Discriminator's Loss was 0.44 and the accuracy of predictions was 76,56%, after 10000 epochs.

That mean that the discriminator did not allow to generator to evolve and train in order to produce more accurate time series.

These results can be visualized by the TableEvaluator library and, as seen at the figure below, they don't match. So, the above results are confirmed.

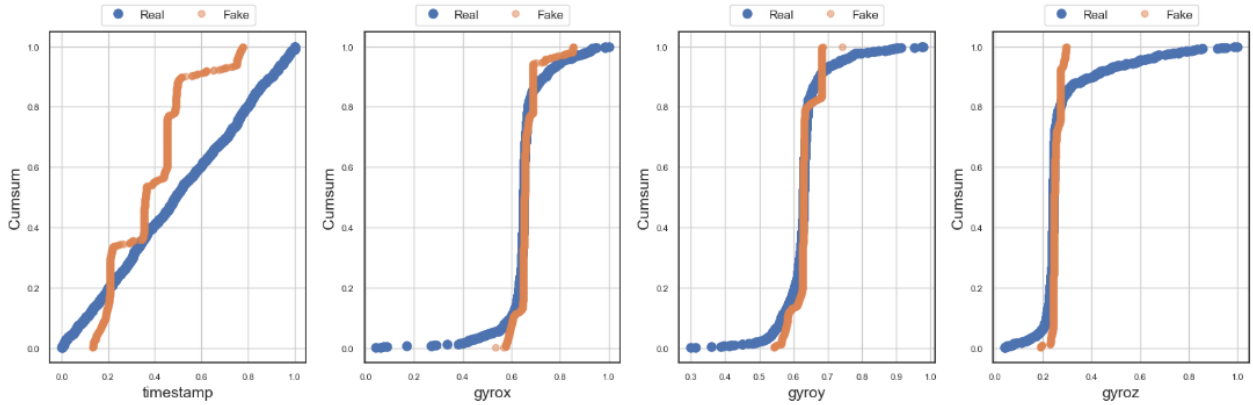


Figure 17:Neural Network GAN cumulative results

4.2.2 GAN creation with LSTM

In the LSTM GAN that both generator and discriminator are created with a mix of BLSTM layers, leaky ReLU and dense layers. A leaky ReLU layer performs a threshold operation, where any input value less than zero is multiplied by a fixed scalar. When the optimizer becomes less fierce when training progresses, some weights may be just too negative and they can no longer

'escape' from the normal ReLU activation, so that death of neural networks can be avoided. The input of the generator is a 4 column (as much as the features are) noise.

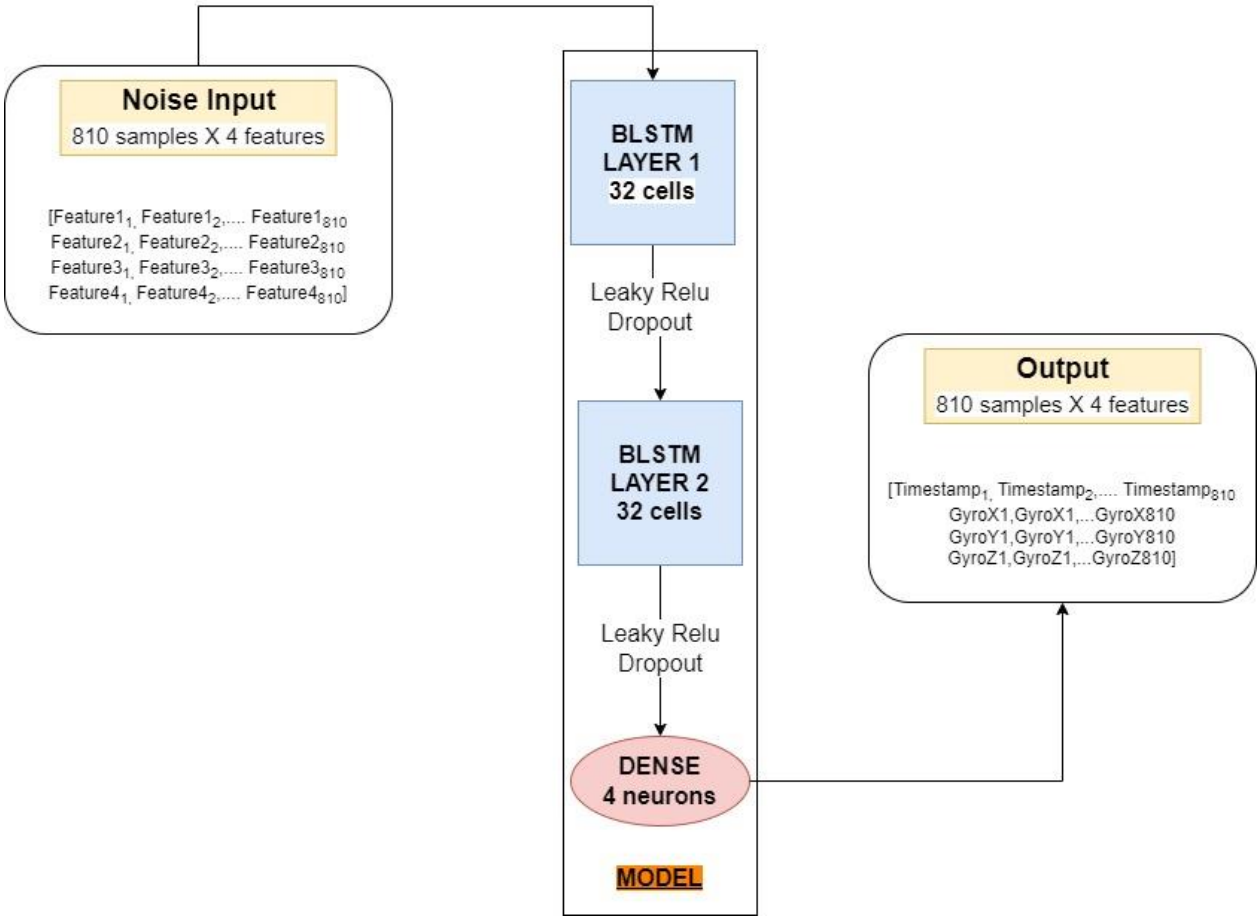


Figure 18:LSTM Generator

Each column consists of 810 random values between 0 and 1. First layer contains, 32 units bidirectional LSTMs, followed by a leaky ReLU unit and a dropout layer. This combination of layers is repeated once more and the generator ends, with a time distributed dense layer which output is a matrix with dimensions equal to 810,4, as much as a given time series is.

The component of the discriminator is 48 units bidirectional LSTMs as first layer, followed by leaky ReLU and time distributed dense layers. In ends with a time distributed dense layer with dimensions of 810,1 as output.

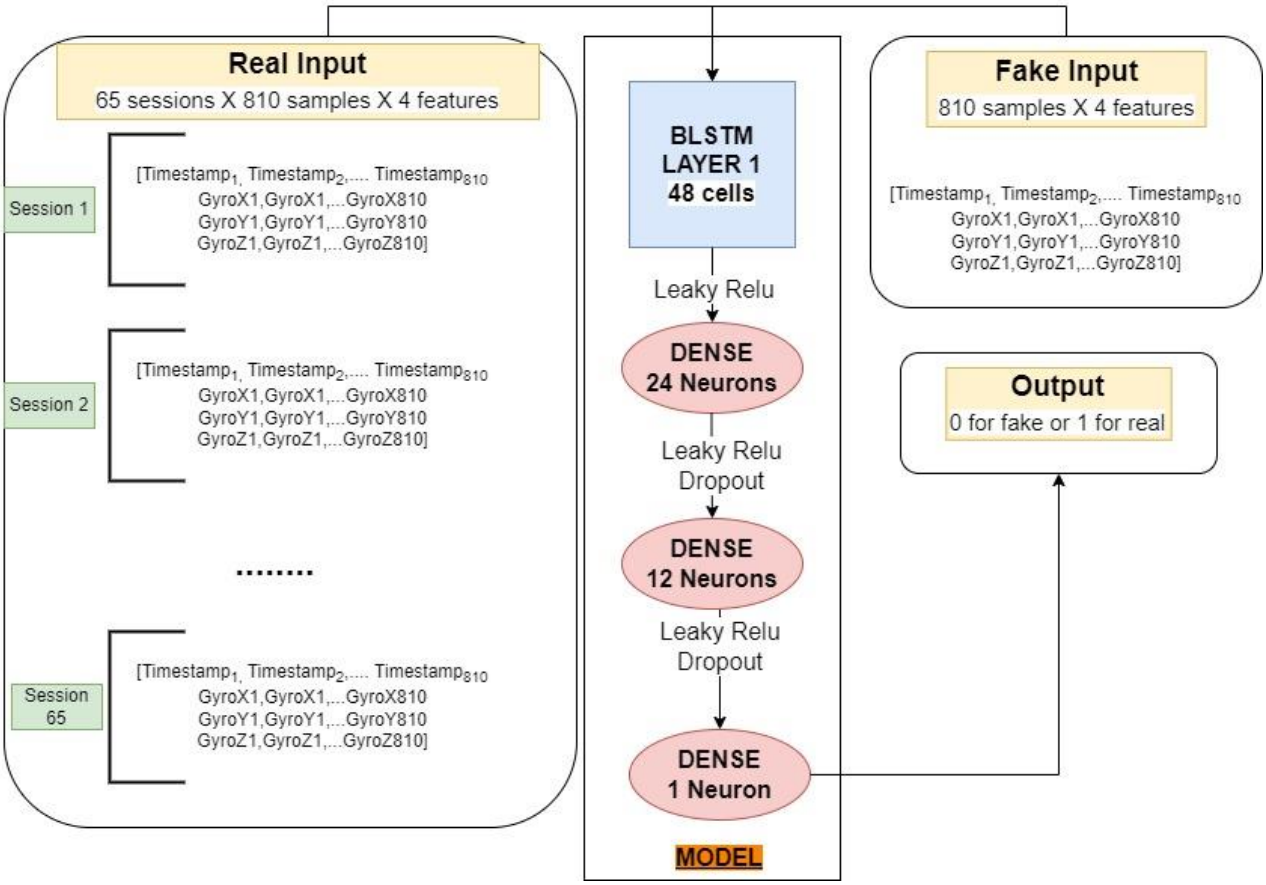


Figure 19:LSTM Discriminator

The discriminator trains manually in every iteration, with two different inputs. First a real one which is a random time series taken by a session of one of the 15 first persons, total 65. Every one of these time series, consists of 810 samples after resampling, with 4 features (3 axis of gyroscope and the timestamp). So real values take the value of 1 as target value. Second a false one, with the same size as the real, which came from generator and taken the value of 0.

The generator trains as follows. By merging the generator and discriminator models into one combined model that works, by taking the output from the generator, which is the fake data, to flow straight into discriminator's input, and right after, calculate the generator and discriminator losses, from the predicted probabilities that comes from the discriminator, in order to recalculate, the weights of the generator model. In other words, the generator attempts to "mock" the discriminator by updating the combination of the two, pointing out that the generated output is a real or valid one. Then the loss for each one of the two models is calculated, the generator is updated by training based by the combination of his and discriminators loss. This model is trained for 10000 epochs with Optimizer is Adam with beta at 0.4, loss is binary_crossentropy and the learning rate was 0.00001.

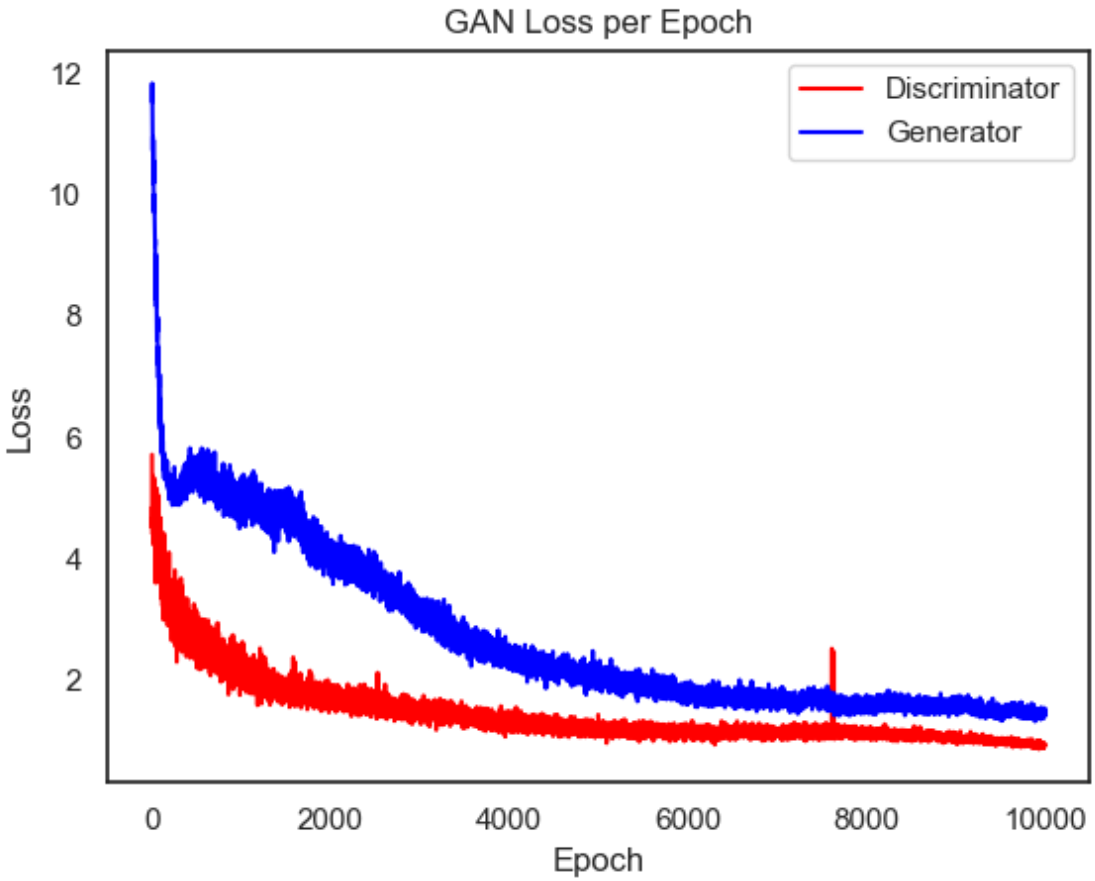


Figure 20:GAN Loss Per epoch

"The plot shows the losses for the discriminator (red) and generator (blue) and clearly shows the steep fall, of both loss values, towards zero over the first iterations, where it remains for the rest of the run" [31]

As seen that the results below the performance of the generator is average.

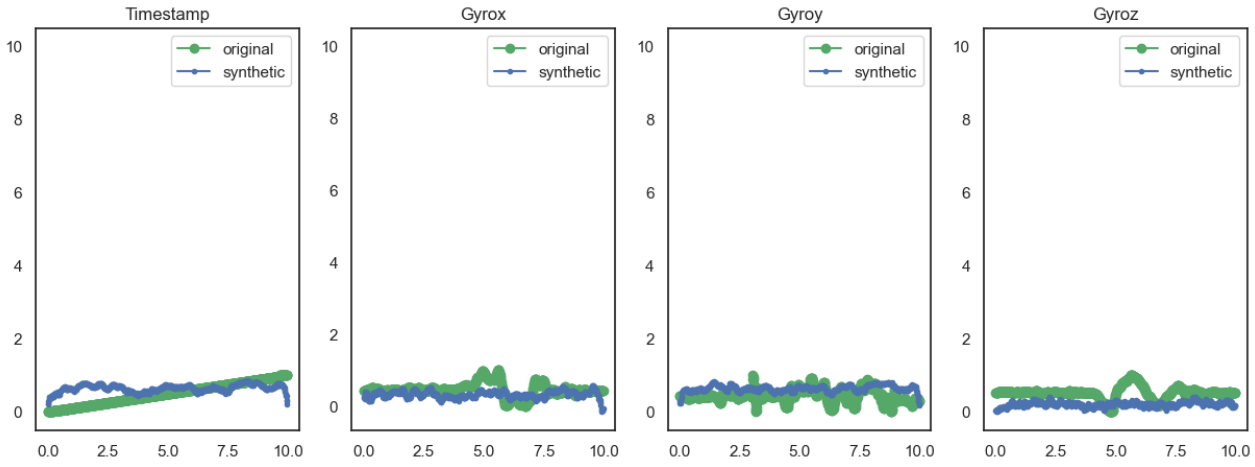


Figure 21:Gyroscope fake data, created from LSTM Generator

4.3 Classification

In conclusion of the experiment, an LSTM network is trained to recognize if the time sequence that is given is generated by the GAN's generator (Bot) or it came directly from the HuMIDB (Human).

4.3.1 Create training and testing Dataset

For the real data were used 183 random user sessions from HuMIDB.

For the fake data were used the two type of Generators (LSTM, Neural Networks) and from different epoch training (9 Generators). The generators used is shown at Table 1.

Type	Epochs Trained
LSTM	Lowest Error
LSTM	4000
LSTM	5000
LSTM	6000
LSTM	8000
NN	4000
NN	5000
NN	6000
NN	10000

Table 1:Generators stored for fake data creation

For all the training dataset were used 65 real Time Series and 65 timeseries Created from Generator, for the total of 130. As for the testing dataset, 118 Real Time Series and 118 timeseries Created from Generator were used for 236 timeseries in total.

4.3.2 Classification models

In this section, a LSTM model for bot classification was created. For the classifier, a simple Biderectional LSTM (64 units) was used as first layer, followed by leaky ReLU and a dropout layer. In ends with a time distributed dense layer with one output, with linear activation function to make 0 or 1 predictions for the two classes (human and bot) in the problem.

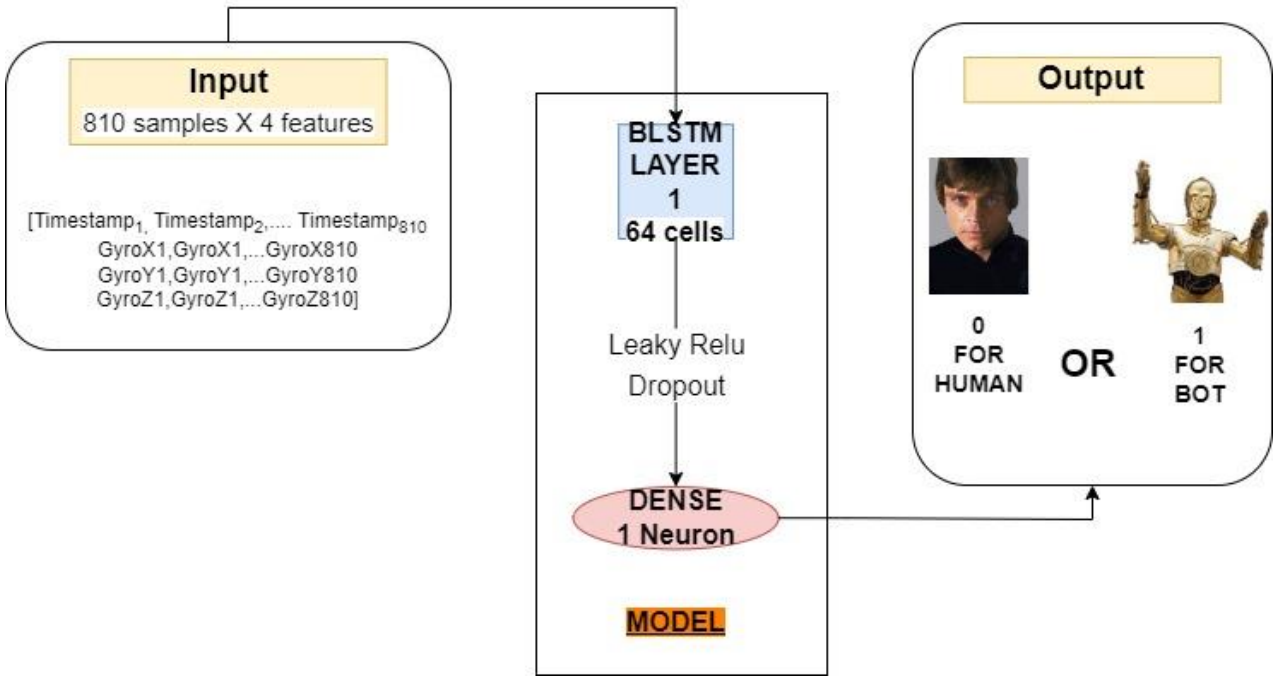


Figure 22:LSTM for classification

Because it is a binary classification problem, binary_crossentropy is used as the loss function. The efficient ADAM optimization algorithm is used. The model is fit for only 10 epochs since it quickly overfits the problem.

Also discriminators (both LSTM and NN) from the GANs training, in different epochs, were stored also, with the intention of, be tested if they can be used as classifiers as well.

4.3.3 Results

All the scenarios were trained with the same size of training (130 Time-Series) and testing (236 Time-Series).

4.3.3.1 Discriminator as classifier

Discriminator testing fake data comes from one LSTM generator only. The discriminators are already trained from the GAN training. So for the different type of discriminators, we get the following results, as seen at the tables below.

Type	Epochs Trained	Accuracy
LSTM	500	50.0
LSTM	1000	50.0
LSTM	1500	50.04
LSTM	2000	50.4

LSTM	2500	51.20
LSTM	3000	47.29
LSTM	3500	32.34
LSTM	4000	25.01
LSTM	4500	23.44
LSTM	5000	25.66
LSTM	5500	23.65
LSTM	6000	22.80
LSTM	6500	33.31
LSTM	7000	45.09
LSTM	7500	49.73
LSTM	8000	50.03
LSTM	8500	50.02
LSTM	9000	50.03
LSTM	9500	50.11
LSTM	10000	50.35

Table 2:LSTM discriminators as classifier

Type	Epochs Trained	Accuracy
NN	200	50.0
NN	300	50.39
NN	400	46.87
NN	500	46.09
NN	1000	46.09
NN	2000	46.09
NN	3000	46.48
NN	4000	46.09

NN	5000	46.48
NN	6000	46.09
NN	7000	46.09
NN	8000	46.09
NN	9000	46.09
NN	10000	46.09

Table 3:ANN discriminators as classifier

The discriminators performance wasn't good enough. Their accuracy was about 50%.

4.3.3.2 LSTM Classifier

LSTM classifier was tested with different versions of training and testing dataset, regarding the fake samples. Everyone of these were evaluated separately.

- ✓ Both training and testing Datasets are created from LSTM Generators
  Excellent Accuracy: **99.70%**
- ✓ Both training and testing Datasets from NN Generators
  Excellent Accuracy: **99.93%**
- ✓ Both training and testing Datasets from Both type of Generators
  Excellent Accuracy: **99.69%**
- ✓ Training Dataset from all Generators and testing from NN Generators
  Excellent Accuracy: **99.68%**
- ✓ Training Dataset from NN Generator and testing from each Generators

Type	Epochs Trained	Accuracy
LSTM	Lowest Error	54.03
LSTM	4000	49.33
LSTM	5000	50.19
LSTM	6000	51.13
LSTM	8000	50.25
NN	4000	99.99
NN	5000	100
NN	6000	100
NN	10000	100

Table 4:Training Dataset from NN Generator and testing from each Generators Accuracy

The LSTM classifier was trained from an ANN Generator so the Accuracy for the samples that came from ANN Generators is excellent. From the other hand, the model had poor performance for the LSTM Generated samples, almost like the discriminator.

This is a very good result, and very promising for future work. In some cases, the testing evaluation, was 100% accurate. Comparing other papers that used same classifier, with better bot performance, their results were almost similar than this approach. Significantly in [3] the results was 90%, in [5] around 80%-90% and in [70] but in a hybrid solution, it was 99.1% which is a superior performance.

5

Discussion

We found that GANs-generated fake sensor timeseries can be easily detected using LSTM. However, it was not surprising to notice that the DL performed better than discriminators for detecting synthetic timeseries. Also the performance of the GAN wasn't so remarkable.

5.1 GAN performance

In applications, GANs can be both beneficial and disruptive, but, on the other hand, there is always a switch, among their benefits, meaning the obstacles that come with the usage with them. We may divide GAN's difficulties into three categories:

- Collapse mode
- Instability and non-convergence
- Sensitivity to hyperparameters and evaluation measures is exceptional.

Inability to find network convergence is one of the leading causes of failure modes.

"GANs are inherently unstable because they are made up of two networks, each with its own loss function. Diving a little deeper into the question, the Generator loss can lead to GAN instability, which can be the cause of the gradient vanishing problem when the Discriminator can easily distinguish between real and fake samples."[15,20]

"This could be a reference to the notion that local equilibria exist in the non-convex game we're training GANs for, as proposed in an article about GANs convergence and stability. Some solutions to this problem are reversing the target used to calculate the cross-entropy cost or using a gradient penalty to avoid local equilibria."[15,20]

"In most circumstances, the ideal situation in which both networks stable and deliver consistent results is difficult to attain. One reason for this issue, is that as the generator improves with subsequent epochs, the discriminator weakens, because is becoming unable to distinguish between true and fake data. "[15]

The generator can easily deceive the discriminator by finding a specific type of data. It'll keep generating the same data under the premise that the target has been met. The system as a whole may over-optimize for that one form of output. The discriminator has a 50 percent accuracy if the generator succeeds every time, akin to flipping a coin. This puts the GAN's overall convergence in

jeopardy. This is a problem with a high degree of unpredictability. It wasn't anticipated until someone observed that the generator produces, just one or otherwise, a limited subset of possible outputs or modes.

"The discriminator, on the other hand, is performing admirably. Because the generator loss is increasing, the outcomes are becoming increasingly terrible, making it very easy for the discriminator to categorize them as false. The loss graph, too, does not reach equilibrium."

The challenge with detecting mode collapse and other failure modes is that qualitative analysis is insufficient (like manually looking at the data). If there is a large amount of data or the problem is really complex, this method may fail. Convergence failure, in practice, means that a visually awful set of data is generated. [15]

"The issue is how to identify and correct failure modes. The first piece of advise is to look at loss graphs. It's also a problem if a loss graph for both the generator and the discriminator declines to zero, in the first epochs. It suggests the generator has discovered a group of false timeseries that the discriminator can easily detect. Figure 20 describes this precise scenario."[20]

"Because the model parameters, fluctuating at a large amount and never converge, some researchers have developed, new loss functions to help GANs find a better optimum. Researchers are likely to experiment with numerous cost functions, before coming up with a final solution for the importance, and they have pointed out that choosing the proper loss function can successfully deal with training instability."[20]

Improvements and innovations, in the field of the ML models, loss functions, include the application of new probability distance and divergence, which can solve and explain with details, the mode collapse problem, while stabilizing GAN training, is proposed like WGAN. Also the WGAN-GP, introduces of Regularization or Gradient Penalty, presenting an improved version of WGAN providing more stable training." EBGAN/BEGAN, LSGAN, RGAN, and RaGAN are some additional typical cost functions utilized in state-of-the-art GANs. There will be more and more, and the existing ones will be upgraded and become more stable, because cost function is an important research field in GANs [20,24,29].

The input, which is usually random noise for the generator, is sampled from latent space. If the latent space is limited, more outputs of the same type will be produced.

Furthermore, a high learning rate has been recognized as one of the most typical challenges encountered while training GANs. Either mode collapse or non-convergence occurs as a result. The learning rate should be kept low, as low as 0.0002 or even lower. The issue that can arise is that due to a high learning rate, the discriminator cannot be educated even a little.

Likewise, an aggressive modifier is detrimental to GAN training. As a result, it is impossible to find a balance between generator loss and discriminator loss, resulting in convergence failure. Betas are the hyperparameters utilized in Adam Optimizer to calculate the gradient's running average and square. Beta = 0.9, which is the default value in many ML libraries, makes the optimizer more aggressive, therefore it needs to be reduced.

Some researches of regularization for improving GAN convergence, suggesting, to add noise to discriminator inputs [10] or/and penalizing discriminator weights [55].

As a result, experiment tracking is critical for training GANs. It's critical to comprehend the loss graphs and pay close attention to the generated intermediate data. If not calibrated properly, hyperparameters such as learning rate, optimizer parameters, latent space, and so on might destroy a GAN model.

With the rise in GAN models in recent years, there has been an increase in study into stabilizing GAN training. There are plenty of other strategies that are useful in specific situations. [6,20,30,53,61].

5.2 Discriminator as Classifier

“The generator samples random noise and provides an output from it as it is being taught. The output is then passed via the discriminator, which classifies it as "True" or "Fake" based on the discriminator's ability to distinguish one from the other. The generator loss is then determined using the discriminator's categorization; if it successfully fools the discriminator, it is rewarded, otherwise, it is penalized. Likewise, the discriminator penalizes itself for misclassifying a true instance as fake or a fake instance, as true”[24]

Typically, the GAN's main goal is to generate a variety of outputs. Instead, the network teaches itself, to predict a specific data distribution through recurrent training, resulting in a consistent output. Every time, the generator is trying to select the one output that appears most credible to the discriminator during the training process. Instead, through repeated training, the network learns to predict a given data distribution, resulting in a reliable output. During the training of the network, the generator is trying to select the one output that appears most believable in order to fool the discriminator.

“Concluding, the best that the discriminatorcan do as a strategy, is the rejection of the generator's output, all of the time. If the next generation or output, of discriminator, remains in a local minima and can't find a way out by recalculate the weights, the next generator training epoch, will have an easier time generating the most feasible output, so a biggest challenge, for the current discriminator. ”[20,24]

5.3 LSTM Classification

Models of LSTM networks are a sort of recurrent neural network that can learn and remember extended sequences of input data. They're designed to work with data that's made up of extended data sequences.

Multiple sequences of input data for example the time series produced by a sensor, such as the axis of the accelerometer and in the current experiment, the gyroscope data, can be supported by the model. The model learns how to extract features from observation sequences and how to link internal features to various activity kinds.

The main advantage of using LSTMs for sequence data or timeseries, classification is that they can learn directly from raw time series data, so they do not, need to construct manually, input characteristics. The model can learn an internal representation of the time series data and accomplish performances that are comparable to models fit on a version of the dataset with engineered features [16,22,30,31,53].

According to the review and the results of the studies, the LSTM classifier should produce good results, which it did. In several instances, the accuracy was nearly 100 percent. The synthetic data technique failed miserably, whereas the simple LSTM functioned admirably.

6

Conclusion and Future Work

The fast adoption of new technology is boosting productivity but also posing new cybersecurity challenges. Identity theft and online account hacking are no longer the exclusive types of cyberattacks. They endanger the physical world, including houses, cities, infrastructure, and medical equipment in people's bodies.

A slew of digital technologies, including artificial intelligence (AI), automated botnets, the Internet of Things (IoT), and cloud computing, both assist and defend attacks on a scale, speed, and sophistication never seen before.

Bots are designed to mimic or replace human user behavior. They are much faster than human users because they are automated. Bots are used by businesses, people and even other AI programs, for undertaking repetitive tasks that would otherwise cannot be performed or difficultly done, by humans. Compared to similar human work, bot tasks are often simple and can be completed at a significantly faster rate, making human life much easier.

Bots come in a variety of forms, including chatbots, social bots, shop bots, and internet bots, which are also known as spiders, crawlers, or web bots. Bots are occasionally employed for criminal actions such as data theft, frauds, or DDoS assaults. They perform useful duties such as customer support or indexing search engines. Attackers may spread malicious bots in a botnet to carry out these attacks and hide the source of the attack traffic.

Bot generation and detection have both been enlisted the help of advanced AI ideas such as LSTM and GANs. The essential concept of a GAN is indirect training via a discriminator, which is another neural network that can determine how realistic an input is and is updated constantly as well. This essentially means that the generator is trained to deceive the discriminator rather than minimize the distance to a certain amount, meaning that the model learns, without supervision [30]. A cell, an input gate, an output gate and a forget gate make up a typical LSTM unit. The three gates control the flow of information into and out of the cell, and the cell remembers values across arbitrary time intervals [13]. Time series data is well-suited to LSTM networks for classification, processing, and prediction.

So, in this thesis two GANs, one made by NNs and one by BLSTM were used for generating bots and a BLSTM for the bot detection. Also there have been a try for the GAN discriminators as classifier.

For the experiment first step was the data preparation, in which took place, the data selection, normalization, resample and standardization. As Dataset the HuMIdb was chosen. Then GANs were trained, and after the detection took place, trying to separate if the sample was given was came from the generators or directly from the dataset

The experiment has acted good in detection, but poor in creation. The problem for the GAN was the convergence failure, which means that a bad set of data is getting generated. So, it was easy for the classifiers to make the right choice. The accuracy of LSTM classifier was in some scenarios 100%. Discriminators were tested as classifiers as well, but the results were 50%. This was also be owed to converge failure.

The hyperparameter adjustment looks to be the solution to the issue. Model hyperparameters such as the number of units, training epochs, batch size, optimizer, loss functions, and learning rate can all be tuned. After the Generator has performed successfully, it is worthwhile to try the discriminator as a classifier once more.

More sensors should be added to the dataset as the experiment progresses, so that more information regarding authentication, or the data series combination that is required for a certain activity, is available.

Finally, we can add more LSTM variations, GRUs, and CONV layers to the classifier and GAN. A hybrid classifier can be utilized, with GANs assisting in the extraction of features in a more detailed fashion.

Because we are living in the Internet of Things era, we must be worried about the potential of good bots as well as the threat of malevolent bots. Bot evolution will be more effective with these methodologies for generating, detecting, and categorizing bots. Many researchers from around the world have put a lot of effort into fixing this problem, and the application of the recommended solutions will lead to increased trust and fairness.

References

[1] Abadi, Martin, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J,others. (2016). Tensorflow: A system for large-scale machine learning. In 12th $USENIX$ Symposium on Operating Systems Design and Implementation ($OSDI$ 16) (pp. 265–283).

[2] Abu-El-Rub N, Mueen A (2019) Botcamp: Bot-driven interactions in social campaigns. In: The world wide web conference, pp 2529–2535

[3] Acien A., A. Morales, J. Fierrez, R. Vera-Rodriguez, Ivan Bartolome, "BeCAPTCHA: Detecting Human Behavior in Smartphone Interaction using Multiple Inbuilt Sensors", In AAAI Workshop on Artificial for Cyber Security (AICS), Feb. 2020.

[4] Acien A., A. Morales, R. Vera-Rodriguez and J. Fierrez, "Smartphone Sensors for Modeling Human-Computer Interaction: General Outlook and Research Datasets for User Authentication," 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), 2020, pp. 1273-1278, doi: 10.1109/COMPSAC48688.2020.00-81.

[5] Acien, A., Morales, A., Fierrez, J., Vera-Rodriguez, R., & Delgado-Mohatar, O. (2021). BeCAPTCHA: Behavioral bot detection using touchscreen and mobile sensors benchmarked on HuMIdb. Engineering Applications of Artificial Intelligence, 98, 104058.

[6] Agrawal, T. (2022, March 21). GANs Failure Modes: How to Identify and Monitor Them. Neptune.Ai. https://neptune.ai/blog/gan-failure-modes

[7] Alarifi A, Alsaleh M, Al-Salman AM (2016) Twitter turing test: identifying social machines. Inform Sci 372:332–346

[8] Alauthman, M., Aslam, N., Al-Kasassbeh, M., Khan, S., Al-Qerem, A., & Choo, K. K. R. (2020). An efficient reinforcement learning-based Botnet detection approach. Journal of Network and Computer Applications, 150, 102479.

[9] Andriotis P, Atsuhiro T (2018) Emotional bots: content-based spammer detection on social media. In: 2018 IEEE international workshop on information forensics and security (WIFS). IEEE, pp 1–8

[10] Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862.

[11] Bahşi, H., Nõmm, S., & La Torre, F. B. (2018, November). Dimensionality reduction for machine learning based iot botnet detection. In 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV) (pp. 1857-1862). IEEE.

[12] Bauke Brenninkmeijer. (2020). Table Evaluator. https://baukebrenninkmeijer.github.io/table-evaluator/.

[13] Berk, M. (2021, August 11). How to learn long-term trends with LSTM. Medium. Retrieved June 1, 2022, from https://towardsdatascience.com/how-to-learn-long-term-trends-with-lstm-c992d32d73be

[14] Beskow DM, Carley KM (2019) Its all in a name: detecting and labeling bots by their name. Comput Math Organ Theory 25(1):24–35

[15] Boldo, S., & Melquiond, G. (2017). 9 - Real and Numerical Analysis. Στο S. Boldo & G. Melquiond, Floating-Point Algorithms and Formal Proofs (σσ. 259–287). doi:10.1016/B978-1-78548-112-3.50009-6

[16] Brownlee, J. (2020, September 3). Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras. Machine Learning Mastery. https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/

[17] Chen S, Y. Chen and W. Tzeng, "Effective Botnet Detection Through Neural Networks on Convolutional Features," 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 2018, pp. 372-378, doi: 10.1109/TrustCom/BigDataSE.2018.00062.

[18] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[19] Chung, Yeounoh & Park, Chang-Yong & Kim, Noo-Ri & Cho, Hana & Yoon, Taebok & Lee, Hunjoo & Lee, Jee-Hyong. (2013). A Behavior Analysis-Based Game Bot Detection Approach Considering Various Play Styles. ETRI Journal. 35. 10.4218/etrij.13.2013.0049.

[20] Clemente, F. (2021, December 15). What is going on with my GAN? - Part 1 | Towards Data Science. Medium. https://towardsdatascience.com/what-is-going-on-with-my-gan-13a00b88519e

[21] Daouadi KE, Rebaï RZ, Amous I (2019) Bot detection on online social networks using deep forest. In: Computer science on-line conference. Springer, pp 307–315

[22] Do, T. (2022, May 5). Time Series Classification Tutorial: Combining Static and Sequential Feature Modeling using Recurrent Neural Networks. Omdena | Building AI Solutions for Real-World Problems. https://omdena.com/blog/time-series-classification-model-tutorial/

[23] Dorri A, Abadi M, Dadfarnia M (2018) Socialbothunter: Botnet detection in twitter-like social networking services using semi-supervised collective classification. In: 2018 IEEE 16th international conference on dependable, autonomic and secure computing, In: 16th International conference on pervasive intelligence and computing, In: 4th International conference on big data intelligence and computing and cyber science and technology congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, pp 496–503

[24] Dwivedi, H. (2022, March 21). Understanding GAN Loss Functions. Neptune.Ai. https://neptune.ai/blog/gan-loss-functions

[25] Ferrara E (2017) Disinformation and social bot operations in the run up to the 2017 french presidential election. arXiv preprint arXiv:1707.00086

[26] García González, G, Casas, P, Fernández, A y Gómez, G. (2020.). Network anomaly detection with Net-GAN, a generative adversarial network for analysis of multivariate time-series. EN: ACM Special Interest Group on Data Communication (SIGCOMM '20 Demos and Posters), Nueva York, NY, USA, 10-14 aug, page 1-3. 3 p.

[27] GeeksforGeeks. (2021, June 25). Understanding of LSTM Networks. https://www.geeksforgeeks.org/understanding-of-lstm-networks/

[28] Gilani Z, Kochmar E, Crowcroft J (2017) Classification of twitter accounts into automated agents and human users. In: Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining, pp 489–496

[29] GitHub -(2021). hwalsuklee/tensorflow-generative-model-collections: Collection of generative models in Tensorflow. GitHub. https://github.com/hwalsuklee/tensorflow-generative-model-collections

[30] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In Advances in Neural Information Processing Systems, pp. 2672-2680. 2014. https://arxiv.org/abs/1406.2661v1

[31] Goodfellow, Ian. (2016). NIPS 2016 Tutorial: Generative Adversarial Networks.

[32] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020).

[33] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

[34] Iliou, Christos & Kostoulas, Theodoros & Tsikrika, Theodora & Katos, Vasilios & Vrochidis, Stefanos & Kompatsiaris, Ioannis. (2019). Towards a framework for detecting advanced Web bots. 1-10. 10.1145/3339252.3339267.

[35] Iqbal, T., & Qureshi, S. (2020). The survey: Text generation models in deep learning. Journal of King Saud University - Computer and Information Sciences. doi:10.1016/j.jksuci.2020.04.001

[36] Khan, Riaz & Zhang, Xiaosong & Kumar, Rajesh & Sharif, Abubakar & Amiri Golilarz, Noorbakhsh & Alazab, Mamoun. (2019). An Adaptive Multi-Layer Botnet Detection Technique Using Machine Learning Classifiers. Applied Sciences. 9. 2375. 10.3390/app9112375.

[37] Kosmajac, D., & Keselj, V. (2019). Twitter bot detection using diversity measures. In Proceedings of the 3rd International Conference on Natural Language and Speech Processing (pp. 1-8).

[38] Kotenko, Igor & Saenko, Igor & Branitskiy, Alexander. (2018). Framework for Mobile Internet of Things Security Monitoring Based on Big Data Processing and Machine Learning. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2881998.

[39] Kumar A., Lim T.J. (2020) Early Detection of Mirai-Like IoT Bots in Large-Scale Networks through Sub-sampled Packet Traffic Analysis. In: Arai K., Bhatia R. (eds) Advances in Information and Communication. FICC 2019. Lecture Notes in Networks and Systems, vol 70. Springer, Cham. https://doi.org/10.1007/978-3-030-12385-7_58

[40] Letteri, Ivan & Penna, Giuseppe & De Gasperis, Giovanni. (2018). Botnet Detection in Software Defined Networks by Deep Learning Techniques: 10th International Symposium, CSS 2018, Amalfi, Italy, October 29–31, 2018, Proceedings. 10.1007/978-3-030-01689-0_4.

[41] Majumdar P., Singh A., Pandey A., Chaudhary P. (2021) A Deep Learning Approach Against Botnet Attacks to Reduce the Interference Problem of IoT. In: Dash S.S., Das S., Panigrahi

B.K. (eds) Intelligent Computing and Applications. Advances in Intelligent Systems and Computing, vol 1172. Springer, Singapore. https://doi.org/10.1007/978-981-15-5566-4_58

[42] Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., & Tesconi, M. (2019, June). Rtbust: Exploiting temporal patterns for botnet detection on twitter. In Proceedings of the 10th ACM Conference on Web Science (pp. 183-192).

[43] McDermott, Christopher & Majdani, Farzan & Petrovski, Andrei. (2018). Botnet Detection in the Internet of Things using Deep Learning Approaches. 10.1109/IJCNN.2018.8489489.

[44] McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

[45] Michael L. Waskom (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021.

[46] Najari, S., Salehi, M. & Farahbakhsh, R. GANBOT: a GAN-based framework for social bot detection. Soc. Netw. Anal. Min. 12, 4 (2022). https://doi.org/10.1007/s13278-021-00800-9

[47] Nõmm, S., & Bahşi, H. (2018, December). Unsupervised anomaly based botnet detection in IoT networks. In 2018 17th IEEE international conference on machine learning and applications (ICMLA) (pp. 1048-1053). IEEE.

[48] Nuha Albadi, Maram Kurdi, and Shivakant Mishra. 2019. Hateful People or Hateful Bots? Detection and Characterization of Bots Spreading Religious Hatred in Arabic Social Media. Proc. ACM Hum.-Comput. Interact. 3, CSCW, Article 61 (November 2019), 25 pages. DOI:https://doi.org/10.1145/3359163.

[49] Papaioannou M., G. Mantas, A. Essop, P. Cox, I. E. Otung and J. Rodriguez, "Risk-Based Adaptive User Authentication for Mobile Passenger ID Devices for Land/Sea Border Control," 2021 IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2021, pp. 1-6, doi: 10.1109/CAMAD52502.2021.9617802.

[50] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[51] Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543

[52] Popoola, Segun & Adebisi, Bamidele & Hammoudeh, Mohammad & Gui, Guan & Gacanin, Haris. (2020). Hybrid Deep Learning for Botnet Attack Detection in the Internet of Things Networks. IEEE Internet of Things Journal. PP. 2327-4662. 10.1109/JIOT.2020.3034156.

[53] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

[54] Rossetti, M., & Zaman, T. (2022). Bots, Disinformation, and the First Trump Impeachment. arXiv preprint arXiv:2204.08915.

[55] Roth, K., Lucchi, A., Nowozin, S., & Hofmann, T. (2017). Stabilizing training of generative adversarial networks through regularization. Advances in neural information processing systems, 30.

[56] Smith, K. E., & Smith, A. O. (2020). Conditional GAN for timeseries generation. arXiv preprint arXiv:2006.16477.

[57] Staudemeyer, Ralf & Morris, Eric. (2019). Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks.

[58] Stragapede, G., Vera-Rodriguez, R., Tolosana, R., Morales, A., Acien, A., & Le Lan, G. (2022). Mobile behavioral biometrics for passive authentication. Pattern Recognition Letters, 157, 35-41.

[59] Tondak, akshay. (n.d.). Recurrent neural networks (RNN) tutorial: RNN training, Advantages & disadvantages (complete guidance). Cloud Training Program. Retrieved June 1, 2022, from https://k21academy.com/datascience/machine-learning/recurrent-neural-networks/

[60] Turing AM (1950) Computing machinery and intelligence. Mind 59(236):433

[61] Understanding Generative Adversarial Networks. (2017, March 5). Seita's Place. https://danieltakeshi.github.io/2017/03/05/understanding-generative-adversarial-networks/

[62] Valliyammai C, Devakunchari R (2019) Distributed and scalable sybil identification based on nearest neighbour approximation using big data analysis techniques. Cluster Computing 22(6):14461–14476

[63] Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

[64] Varol O, Ferrara E, Davis CA, Menczer F, Flammini A (2017) Online human-bot interactions: detection, estimation, and characterization. In: Eleventh international AAAI conference on web and social media

[65] Verma, Y. (2021, November 20). Complete Guide To Bidirectional LSTM (With Python Codes). Analytics India Magazine. https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/

[66] Vinayakumar Ravi, & Alazab, Mamoun & Srinivasan, Sriram & Pham, Quoc-Viet & Padannayil, Soman & Ketha, Simran. (2020). A Visualized Botnet Detection System Based Deep Learning for the Internet of Things Networks of Smart Cities. IEEE Transactions on Industry Applications. 1-1. 10.1109/TIA.2020.2971952.

[67] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

[68] Wang, Y., & Wang, L. (2019, May). Bot-like Behavior Detection in Online Banking. In Proceedings of the 2019 4th International Conference on Big Data and Computing (pp. 140-144).

[69] Wu, D., Fang, B., Wang, J., Liu, Q., & Cui, X. (2019, May). Evading machine learning botnet detection models via deep reinforcement learning. In ICC 2019-2019 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.

[70] Xu X., H. Zhao, H. Liu and H. Sun, "LSTM-GAN-XGBOOST Based Anomaly Detection Algorithm for Time Series Data," 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), 2020, pp. 334-339, doi: 10.1109/PHM-Jinan48558.2020.00066.

[71] Yin C., Y. Zhu, S. Liu, J. Fei and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), 2018, pp. 228-234, doi: 10.1109/ICAIBD.2018.8396200.

[72] Yu L, Zhang W, Wang J, Yu Y (2017) Seqgan: aequence generative adversarial nets with policy gradient. In: Thirty-first AAAI conference on artificial intelligence

[73] Zhao J, Liu X, Yan Q, Li B, Shao M, Peng H (2020) Multi-attributed heterogeneous graph convolutional network for bot detection. Inform Sci 537:380–393