IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 13, 2018, accepted October 3, 2018, date of publication October 15, 2018, date of current version November 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2875068

Mobile Forensic Data Analysis: Suspicious Pattern Detection in Mobile Evidence

KONSTANTIA BARMPATSALOU, TIAGO CRUZ, (Senior Member, IEEE), EDMUNDO MONTEIRO, (Senior Member, IEEE), AND PAULO SIMOES, (Member, IEEE)
Centre for Informatics and Systems, University of Coimbra, 3030-290 Coimbra, Portugal

Corresponding author: Konstantia Barmpatsalou (konstantia@dei.uc.pt)

This work was supported by the ATENA H2020 EU Project (H2020-DS-2015-1 Project 700581).

ABSTRACT Culprits' identification by the means of suspicious pattern detection techniques from mobile device data is one of the most important aims of the mobile forensic data analysis. When criminal activities are related to entirely automated procedures such as malware propagation, predicting the corresponding behavior is a rather achievable task. However, when human behavior is involved, such as in cases of traditional crimes, prediction and detection become more compelling. This paper introduces a combined criminal profiling and suspicious pattern detection methodology for two criminal activities with moderate to the heavy involvement of mobile devices, cyberbullying and low-level drug dealing. Neural and Neurofuzzy techniques are applied on a hybrid original and simulated dataset. The respective performance results are measured and presented, the optimal technique is selected, and the scenarios are re-run on an actual dataset for additional testing and verification.

INDEX TERMS Mobile forensics, evidence data analysis, criminal profiling, behavioral evidence analysis, neural networks, ANFIS.

I. INTRODUCTION

Mobile *Forensic Data Analysis (MFDA)* is one of the least developed, but also one of the youngest *Mobile Forensics (MF)* subdisciplines. As mentioned by Rogers [56], digital forensic research has "fallen into the trap of focusing almost exclusively on the collection of data and has paid very little attention to the examination and analysis phases." The need for implementation of intelligent solutions that will cast off the burden of manual investigations has been highlighted by a respectful amount of research papers throughout literature [6], [37], [49], [55]. Moreover, a questionnaire–based survey by Al Fahdi *et al.* [1] concerning the contemporary issues faced by forensic practitioners suggests that 85% of the participants annotated the "need to develop approaches to identify and extract significant data through techniques such as criminal profiling" [1] as an important issue.

The role of data and metadata forensic investigation is double. Firstly, it is a failsafe mechanism in case direct access to evidence does not succeed in producing a concrete outcome due to anti-forensic scenarios, such as data cascading or deliberate data alteration, such as encoded verbal communication between criminals. Secondly and more related to the current paper, it can become a means of off-loading investigators' tasks, serving as a triage mechanism for potentially suspicious user behavioral patterns before or after a hands-on investigation.

The current paper is the evolution of our previous work, involving the use of Fuzzy Logic as a suspicious pattern detection method for SMS messages in a *Public Protection and Disaster Relief (PPDR)* environment [4]. Despite the fact that the evaluation results were rather satisfactory [7] and Fuzzy Logic is able to successfully identify potentially suspicious entities, there are still some issues in need of improvement.

Due to their nature, Fuzzy Systems are incapable of maintaining a memory of previous states and, consequently, learning from the data they have already used. This creates a rather big impediment when new datasets are introduced. Moreover, the new input variables may have different ranges, thing that requires a full fuzzy parameter reconfiguration, rendering the solution inflexible for lucid and rapid-changing environments.

Fuzzy Logic is an optimal solution for a relatively small number of inputs and is able to handle a small to average number of membership functions per input variable.

Moreover, they are rather reliable, because, among others, they provide interpretable models based on a series of rules [2]. However, when the number of inputs increases, the number of generated Fuzzy rules expands considerably, thus ''increasing the computational complexity of the system and decreasing its overall comprehensibility and interpretability'' [34]. Type–2 and Type–3 Fuzzy Logic are known solutions to this issue, but the lack of learning capabilities is still present.

On the contrary, *Neural Networks (NNs)* and *Neuro-Fuzzy Systems (NFSs)* are faster and more scalable. While Fuzzy Systems are mainly tools for knowledge representation, NNs and NFSs retain knowledge and use it for learning purposes of future input features. There is no need for manual rule inference and parameters are automatically learnt through the provided data. However, such infrastructures are prone to data over- and under-fitting, thing that can be avoided by an analytical training process and a careful definition of components, such as hidden neurons and layers.

As mentioned earlier, the authors proved that Fuzzy Systems can be efficient as means of suspicious pattern detection in mobile forensic evidence for small-scaled problems. This paper introduces an extended criminal profiling and suspicious pattern detection methodology in calls and SMS datasets. Behavioral profiles are built for cyberbullying and drug-dealing scenarios, two use cases that involve the operation of mobile devices and therefore can lead to the creation of a detailed digital criminal profile. Once the different suspiciousness levels that derive from calls and SMS attribute combinations are defined, the performance of NNs and the *Adaptive Neuro-Fuzzy Inference System (ANFIS)* on their detection is evaluated. Lastly, the most efficient out of the aforementioned techniques is tested anew on previously unknown data from an experimental mobile device with the aid of *Android Data Acquisition and Examination Tool (ADAET)* a script developed for the purpose of the current paper, which is also responsible for the automation of the data acquisition and analysis process on mobile devices. The aforementioned verification procedure serves as an additional layer of integrity over non- or partially interpretable intelligent computation models, such as NNs [41].

The rest of the paper is organized in the following manner. Section II performs a literature review, while Section III presents the methodology used in this paper, by analytically describing every part of the process. Section IV demonstrates the results of the experiments. Finally, Section V discusses the most critical findings and potential improvement and SectionVI concludes the paper.

II. RELATED WORK

There is a considerable amount of research papers that employ intelligent computing methods in order to use *Behavioral Evidence Analysis (BEA) techniques and perform automated criminal profiling. The great majority of them makes use of demographic data and qualitative basis in order to proceed to inference. Additionally, their main goal is the creation or the validity verification* of an already existing knowledge base.

One of the first attempts towards this direction was the NNPCP project [62], which comprised the creation of a NN capable of performing criminal profiling among different crime types. Its data pool of inputs were official criminal records from the Italian police force. Despite the relatively impressive description for the time the paper was written, the description of the NN architecture and functionalities is rather abstract and no further details on produced results are provided.

Ferrari *et al.* [18] used Bayesian and Feed-Forward NNs so as to ''model criminal behavior from post-mortem databases of single-victim homicides'' and compared the results of both solutions. The systems used as inputs different psychosocial factors concerning the offenders' character, as well as the way each crime was conducted and classified different criminal actions to different outputs.

Enache *et al.* [17] designed a multilayer NN that aimed to create a demographical and activity-based cybercriminal profile according to an input set of crime types and their associated activities. The authors claim that some successful associations were made; however, complete results were not presented.

The paper by Islam and Verma [36] used Fuzzy Logic concepts in order to perform a risk assessment on messages exchanged by various entities in a 3G network, depending on their identity and motives. The system inputs comprised the variable combination of the SMS senders' degree of acquaintance to the device owner and the type of device they have been using. The system output was the overall risk per input combination, measured in a scale from zero to five. Zero represented the lowest degree of risk per SMS, whereas five represented the highest.

Lai *et al.* [39] introduced ''a conceptual framework for profiling internet pirates,'' according to their behavioral traits on technology use. Lai *et al.* [39] constructed the internet pirate's profile based on three pillars; ''the facts, the behavioral characteristics and the personality particulars.'' The facts category referred to an amount of various observations inferred by the existing data, such as timestamps and exchanged file types. The behavioral characteristics group incorporated traits concerning a pirate's Internet usage, whereas the personality particulars category comprised more abstract notions, such as personality characteristics, reasons that led to piracy and influences that formed the potential pirate's profile. Afterwards, they created and distributed a questionnaire consisting of content related to the three aforementioned categories. They used the *Multidimensional Scaling (MDS)* [10] methodology, so as to form clusters with correlated characteristics and create the respective piracy profiles.

Andro-AutoPsy is a recently introduced antimalware tool, with an innovating attribute. Apart from the information about the malware technical characteristics, the tool uses ''similarity matching in malware creator-centric information'' [33], so as to construct the respective

criminal profiles. Information concerning the creators behavior is usually encountered in ''.smali opcodes, metadata in the *AndroidManifest.xml* file, as well as in serial numbers of various certificates'' [33]. Andro-AutoPsy is actually a hybrid detection engine, consisting of a rule–based, behavioral detection module and a classification engine that performs comparisons among already existing malware activities and decides upon the suspicion of a sample.

Quick and Choo [52] introduced an extensive process model for intelligent MF, named the *Digital Forensics Analysis Cycle (DFIAC)*. The model was then applied to a procedure of retrieving information from various mobile devices confiscated by the South Australian Police for the time period between 2000 and 2015. The authors were able to successfully establish association links among different criminal entities.

It is rather noticeable that the criminal profiling research has matured overtime. Recent works are more sophisticated and concrete. Moreover, they contain a bigger amount of experiments and results, thus providing stronger proof. Lastly, the authors do not hesitate to combine more than one methodologies and use interdisciplinary notions, so as to achieve better results. One of their shortcomings is that they are either profiling or detection tools. The only exception is Andro-AutoPsy, but it is also more focused on machine behavior. The methodology applied in the current paper is a concatenation of the already existing profiling techniques with a newly introduced suspicious pattern detection technique, based on NNs and ANFIS. The next section presents the adopted methodology in detail.

III. METHODOLOGY

As already mentioned in the previous section, research papers on intelligent criminal profiling and behavioral analysis, in addition to their multi–disciplinary nature, they also tend to adopt and adapt elements from many different methodologies in order to improve the overall quality levels of their outcomes. The current paper is one of these examples. It comprises two main phases, Induction and Identification. The Induction phase is related to the construction of a culpable profile and the respective data management. The Identification phase consists of the NN and ANFIS training and validation, their evaluation and their behavior testing on previously unknown data. Table 1 summarizes the content of the two phases.

TABLE 1. Methodology phases.

Induction	Identification
Use Case Definition	Training
Dataset Selection	Evaluation
Preprocessing	Testing on Unknown Data
Ground Truth Generation	Selection

The methodology core comprises a compilation of different principles and formal methods, which are explained in the following paragraphs.

Behavioral Evidence Analysis

BEA is the process of ''deducing the psychobehavioral portrait of the offender based on professional training and previous investigations'' [18] by a group of experts. However, this approach tended to be highly subjective, so a more consistent solution was needed.

Inductive Analysis

Inductive analysis has a more solid scientific base. The methodology aims to use previously committed crimes and data associated to them, so as to create a profile, the patterns of which are a match to yet to be discovered potential offenders. Data encountered in the crime scene are combined with the already existing theory for the production of hypotheses. The latter ones are then analyzed according to past investigations and expert knowledge in the field. Assumptions lead to the validation or alteration of the hypotheses and the scheme continues evolving according to the newer additions that influence the decisions taken. Fig. 1 presents the inductive analysis procedure.

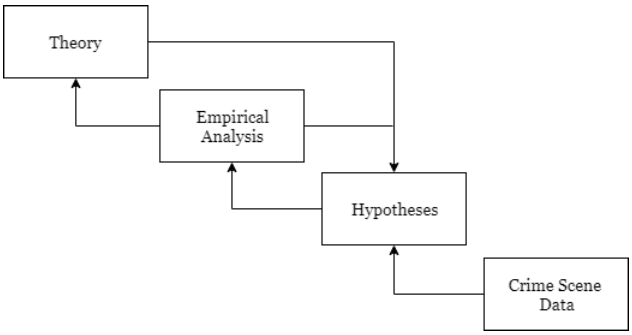


FIGURE 1. Inductive analysis, adapted from [27].

CRISP-DM

CRoss-Industry Standard Process-Data Mining (CRISP-DM) includes the following steps: defining the problem and structuring a solution strategy; accumulating information from seized media and forming the datasets; performing feature selection; ''matching of data in order to identify deficiencies, discrepancies or similarities'' [45]; applying the knowledge retrieved from the data so as to construct a cybercriminal profile and evaluating the validity of the new profile.

The Intelligence Cycle

In 2011, the *United Nations Office on Drugs and Crime (UNODC)* published a guide on generic criminal intelligence analysis, applicable to a variety of cases. One of the basic models introduced in the publication was ''The Intelligence Cycle,'' a circular scheme consisting of seven phases. The first phase, *Tasking*, is related to the crime under examination, the motivations that led to its conduction and the offenders' motivation. The *Collection* phase is a ''formally defined approach to describing the information needed and the means

of acquiring it'' [67], whereas the *Evaluation* phase is responsible for reassuring if the aforementioned information is reliable and in an appropriate state so as to constitute sufficient evidence. *Collation* is responsible for organizing and converting the information to an editable data format. The data are thoroughly examined and important features are highlighted during the *Analysis* phase. The *Inference Development* phase contains all the different types of assumptions that can be produced after the *Analysis* phase. They may be hypothetical, they may concern current or future outcomes, but they can also be concrete and conclusive. Lastly, the *Dissemination* phase concerns the publication of the investigators' findings in electronic or other type of sources.

Rogers' Behavioral Evidence Analysis Model

Rogers [56] introduced a BEA model accustomed to digital investigations. Despite the fact that the model has a linear representation, swapping between phases is applicable. It consists of six phases, namely: *Classification, Context Analysis, Collection, Statistical Analysis, Visualization* and *Decision/Opinion*. *Classification* corresponds to the criminal case selection and the definition of its attributes. *Context Analysis* provides a better understanding of the system under investigation, potential evidence locations and raises the investigator's awareness for the existence of anti–forensic techniques. Similarly to aforementioned models, the *Collection* phase is related to ''evidence collection and storage in a format that can be analyzed for patterns, linkages, and timeline analyses'' [56]. *Statistical Analysis* comprises the methods used in order to detect potentially abnormal occurrences among the evidence. One of the most common techniques used is frequency analysis, but that does not prohibit the investigators from using a broader spectrum of methodologies. *Visualization* is responsible for presenting the key findings in a timeline manner, whereas *Decision/Opinion* produces a final report that contains the conclusions of the evidence analysis.

DFIAC

As already mentioned in Section II, Quick and Choo [52] created a model inspired by the UNODC [67] guidelines. DFIAC is rather similar to *The Intelligence Cycle*. It contains an additional *Prepare* phase, that is responsible for the contextualization of the crime case, a *Collect, Preserve and Collate* phase, which is a variant of the UNODC's equivalent *Collection* with additional pre-processing and its last phase, *Identification of Future Tasks*, an extended version of the *Dissemination* process, including concerns that are yet to be resolved.

It is noticeable that the more concurrent models do not follow a static representation, but have a rather dynamic character. They support internal loops whenever the investigation reveals a new or recurring unresolved concept. Thus, they are rendered more flexible and they provide a higher degree of liberty. The methodology presented in this paper is a concatenation of previously adopted criminal profiling models, enhanced by a suspicious pattern identification routine. Table 2 shows the association between the work presented in the current paper and the aforementioned research papers. The left column comprises the steps of the current methodology, whereas its right equivalent is a part of the already existing models. While the similarity level between the components of the Induction phase and the related literature is considerably high, the Identification phase equivalents cover broader and different scopes, but have similar high level representation. In the following paragraphs, the phases that constitute the proposed methodology are analytically explained.

TABLE 2. Methodology compilation.

Current Methodology	Related Literature
Induction Phase	
Use Case Selection	Problem Definition [45], Classification [56], Tasking [52], [67]
MO Definition	Theory [27], Context Analysis [56], Prepare [52]
Dataset Formation	Collection [45], [56], [67], Crime Scene Data [27], Collect, Preserve, Collate [52]
Variable Definition	Feature Selection [45], Evaluation [52], [67]
Pre–processing	Information Matching [45], Collation [67], Collect, Preserve, Collate [52]
Ground Truth Generation	Building Digital Profile [45], Inference Development [67], Hypotheses [27]
Investigation Phase	
Training	Building Digital Profile [45], Empirical Analysis [27], Statistical Analysis [56]
Evaluation	Evaluate New Profile Validity [45], Visualization [56]
Testing on Unknown Data	Evaluate New Profile Validity [45]
Selection	Decision/Opinion [56], Future Tasks Identified [52]

A. USE CASE DEFINITION

The discipline of *Mobile Forensic Data Analysis (MFDA)* handles acquired artifacts from devices that are either compromised by malicious entities (malware propagation, bitcoin miners, botnet zombies) or serve as means to facilitate the conduction of a crime [8]. The current paper focuses on the latter category, which is the one more strongly correlated to human behavior. More precisely, it aims to pinpoint different criminal activities to specific metadata patterns. The authors selected two offender types for the current examination. The selection was performed according to the public availability of information concerning the offenders' involvement with mobile devices. The authors queried content related to different criminal digital profiles in Psychology, Law and IT journals and the cases that allowed for a higher level of analysis due to their availability and abundance of information were cyberbullying and low-level drug dealing. However, the proposed methodology can be applied with no limitations to whichever criminal profile is involved with mobile devices in a manner that leads to the production of a digital fingerprint of calls and SMSs.

In order to proceed to the detection and correlation procedure, each offender's *Modus Operandi (MO)* has to be defined. Once the characteristics are outlined, rules related to the suspiciousness of data patterns can be inferred and

the ground truth can be generated. Law enforcement has long held to the belief that understanding the methods and techniques criminals use to commit crime is the best way to ''investigate, identify, and ultimately apprehend them'' [66]. The following paragraphs analyze the MO of the two aforementioned offender types.

1) CYBERBULLYING

Bullying by mobile devices seems to be a growing trend and ''was perceived to have a rather negative impact'' [60]. One of the main characteristics of cyberbullies is the very frequent use of their mobile devices, especially for texting via the native or other downloaded applications [19], [28], [40], [54]. However, this fact by itself cannot be considered a framing factor, since teenagers are classified as heavy mobile device users. Cyberbullies tend to send massive text messages [54], [57], [60] and they prefer to ''attack'' their victims after school, especially at night, when their activity is usually not monitored by parental controls [19], [29], [50]. The messages they send are not long; however, they just tend to be annoying by sending small to medium-sized, but insulting texts [50], [54]. Moreover, they also perform many missed or low duration calls to the victims, in order to bother them more or to even provoke them to reply in case they decided to ignore them [54], [60].

A cyberbully's MO shows relatively intense device usage and thus facilitates the inference of a digital fingerprint. The next paragraph highlights a low-level drug dealer's MO.

2) LOW LEVEL DRUG DEALING

Low level drug dealing targets dealers of small quantities, who interact more with potential buyers and less with cartel leaders or other providers. As a result, the majority of their call and message exchanges takes place among entities within the same country [20], [44], [47]. Dealers prefer using mobile devices because they prevent them from increased physical interaction with the clients, which increases the probabilities of being arrested [13], [43]. Drug dealers are highly active in terms of message exchange [16], [20], [47] and call performance [16], [20], [44]. They also interact frequently with specific people, their clientèle, mainly during evenings and nights. Their calls have small duration and they are usually the ones performing than receiving them, based on their convenience. The text messages they send have medium to relatively long length [5], [47] and contain information about the products they are selling, often mixed with irrelevant phrases.

Once the offenders are profiled, their MO can be transformed into a series of encoded actions. For that purpose, the appropriate datasets need to be used. The next subsection describes this procedure, as well as the limitations the authors faced due to limited data availability.

B. DATASET SELECTION

It is a generally accepted truth that ''data are barely shared among the Digital Forensics community'' [30]. Law enforcement agencies have adopted a strict policy about sharing on-field acquired data with the scientific community, so the majority of the publicly available datasets are results of human or computer generated simulations. Digital Forensic datasets are scattered and derive from a variety of sources. Disk images and network dumps are significantly higher in quantity than mobile device images or parts of the device storage.

Most of the existing datasets related to mobile devices, such as the *Computer Forensic Reference Data Sets (CFReDS)* [48] and Digital Corpora [15] correspond either to devices with older mobile OS versions or the available data are not enough in quantity so as to adhere to scenarios which demand repetition of experiments over time. Moreover, datasets relevant to actual criminal investigations are not made publicly available to the community and are under strict authorities' jurisdiction.

The *Cambridge Device Analyzer (CDA)* [68] dataset is a collection of Android usage statistics from various users worldwide, who voluntarily provide their data by installing an application-agent. Access to the full dataset is provided after signing a mutual agreement, where one of the ends is either an academic entity or an organization. The data are sorted by the device they were acquired from and each part contains information collected over a period of six months. All the usage statistics are stored in a *Comma Separated Value (.csv)* file and are formatted according to a scheme containing a unique numeric identifier, a timestamp, a label of the data type and a string field of the corresponding attributes. A more detailed representation of a data tuple can be found below:

$$\texttt{tuple} = \{\texttt{id.}, \texttt{timestamp}, \texttt{type}, [\texttt{attr.}_1, \texttt{attr.}_2, .., \texttt{attr.}_n]\} \quad (1)$$

The dataset used in the current paper comprises joined data from three different devices, resulting in a total of 2,591 call and 11,989 SMS patterns. The data tuple in Equation 1 is split in such a way that each column will belong to a unique attribute. Not every piece of information is useful for the research, so data are filtered and redundancies are removed. The data types used are calls and SMS with their respective attributes. However, their format is not in the appropriate state to be properly interpreted by a NN or a NFS perceptron. This can be achieved by applying a pre-processing procedure, which is described analytically in the following section.

C. PRE-PROCESSING

Pre-processing is related to any data modification that can facilitate their interpretation by the NN perceptrons and ANFIS. Continuous numeric variables do not need further alteration. Linguistic variables that describe different states or numeric variables that denote time frames are translated into numeric discrete values, following the categorization pattern adopted by Barmpatsalou *et al.* [7]. Lastly, linguistic or date-related variables that are not useful for the investigation in their current formatting are transformed into a summation

of their appearance instances. The initial format for both calls and SMS data types used in this research can be found below and a more detailed explanation follows in the next paragraphs.

$$\texttt{Call(type, timestamp, name, location, number\_type, duration)} \quad (2)$$

$$\texttt{SMS(type, timestamp, name, location, number\_type, length)} \quad (3)$$

1) COMMON ATTRIBUTES

Both the Calls and the SMS datasets have some equal attributes, the pre-processing procedure of which is going to be explained in a common space.

a: NAME

The *Name* attribute corresponds to the name or the phone number of the individual with whom the owner of the device interacted. All the names and numbers in the CDA dataset are anonymized and thus, no sensitive information can be extracted from their raw format. However, the instances of each number lead to the creation of *Appearance Frequency*, a variable concerning the amount of total owner interaction with various other entities by calls or text messages.

b: TIMESTAMP

Timestamp is a unified string, comprising the date and the time a call was performed or an SMS was sent or received. This string is later split into the *Date* and *Time* attributes. Despite the fact that the date itself is a useful observation in terms of a digital investigation, it does not provide useful insights for the scope of the current research. Thus, it is converted to the *Daily Frequency* variable, which is the amount of interactions a user had within 24 hours. The *Time* variable is converted to four discrete categories, according to Table 3.

TABLE 3. Time quantification.

Time of the day	Value
Morning (05:01-12:00)	0
Afternoon (12:01-17:00)	1
Evening (17:01-22:00)	2
Night (22:01-05:00)	3

c: LOCATION

The *Location* attribute is represented by linguistic terms in the CDA dataset and refers to whether the phone number of an entity that interacted with the device owner is foreign, local and unknown or undefined, due to parsing errors. The generated *Country Code* variable has three discrete values that are presented in Table 4.

d: NUMBER TYPE

Similarly to the *Location* attribute, *Number Type* consists of strings that describe if the number the user is interacting is mobile, unknown or a fixed line. The generated *Mobility* variable is also present in Table 4.

TABLE 4. Country code and mobility quantification.

Location	Value	Number Type	Value
Foreign	0	Fixed Line	0
Unknown/Undefined	1	Unknown/Undefined	1
Local	2	Mobile	2

Apart from the common data attributes, there are also two more data categories that correspond exclusively to the Calls and SMS types and are described in the following paragraphs.

2) CALL-EXCLUSIVE ATTRIBUTES

The calls data type comprises two attributes that are unique and create two different variables.

a: TYPE

The call *Type* is a binary variable and receives the value *0* for outgoing and *1* for incoming calls.

b: DURATION

Duration is also a call-specific continuous variable, which receives positive integer values in seconds. For the missed calls, the value *–1* is assigned. Zero could also be an assigned value for a missed call, but after a careful observation of the original dataset, there were some incoming and outgoing calls with very small duration that received the same value.

3) SMS-EXCLUSIVE ATTRIBUTES

Similarly to the calls, there are also two variables dedicated to the SMS texts.

a: TYPE

The SMS *Type* is a binary variable and receives the value *0* for sent and *1* for received SMS messages.

b: LENGTH

The last SMS-specific attribute, *Length*, is a continuous variable, receives positive integer values and corresponds to the total number of characters in a text message.

In the end of the pre–processing procedure, both calls and SMS data types consist of seven variables and are fully quantified. Thus, they are ready to be used as inputs for the phases of the ground truth generation and the NN and NFS perceptron training and testing.

D. GROUND TRUTH GENERATION

Barmpatsalou *et al.* [7] introduced an alternative representation of the output suspiciousness. Instead of using the classic binary format (0: not suspicious - 1: suspicious), the output is a fuzzy variable, receiving values within the [0,1] interval. Values closer to zero are considered innocent, whereas values

Algorithm 1 Calls Preprocessing

```
procedure Pre-Processing
    function splitAttributes(RawDataset) return SplittedDataset[number_of _attributes]
    end function
    function createCallType(SplittedDataset[type])
        t ← list()
        for each line in SplittedDataset[type] do
            if type = incoming then
                t ← 1
            else if type = outgoing then
                t ← 0
            end if
        end forreturn t
    end function
    function createCallTime(SplittedDataset[time])
        tm ← list()
        for each line in SplittedDataset[time] do
            if time ≤ 12 : 00 and time > 05 : 00 then
                tm ← 0
            else if time ≤ 17 : 00 and time > 12 : 00 then
                tm ← 1
            else if time ≤ 22 : 00 and time > 17 : 00 then
                tm ← 2
            else if time ≤ 05 : 00 or time > 22 : 00 then
                tm ← 3
            end if
        end forreturn tm
    end function
    function createCallDate(SplittedDataset[date])
        d ← list()
        for each line in SplittedDataset[date] do
            d ← count(date)
        end forreturn d
    end function
    function createCallAf(SplittedDataset[name])
        a ← list()
        for each line in SplittedDataset[name] do
            a ← count(name)
        end forreturn a
    end function
    function createCallCnt(SplittedDataset[place])
        cc ← list()
        for each line in SplittedDataset[place] do
            if place = local then
                cc ← 2
            else if place = unknown then
                cc ← 1
            else if place = foreign then
                cc ← 0
            end if
        end forreturn cc
    end function
    function createCallMob(SplittedDataset[mobility])
        mob ← list()
        for each line in SplittedDataset[mobility] do
            if mobility = mobile then
                mob ← 2
            else if place = unknown then
                mob ← 1
            else if place = fixedline then
                mob ← 0
            end if
        end forreturn mob
    end function
    function createCallDr(SplittedDataset[duration])
        dr ← list()
        for each line in SplittedDataset[duration] do
            dr ← duration
        end forreturn dr
    end function
    function createNNInp(t, tm, d, a, cc, mob, dr)
        join(t, tm, d, a, cc, mob, dr) return SMSInputs
    end function
end procedure
```

Algorithm 2 SMS Preprocessing

```
procedure Pre-Processing
    function splitAttributes(RawDataset) return SplittedDataset[number_of _attributes]
    end function
    function createSMSType(SplittedDataset[type])
        t ← list()
        for each line in SplittedDataset[type] do
            if type = received then
                t ← 1
            else if type = sent then
                t ← 0
            end if
        end forreturn t
    end function
    function createSMSTime(SplittedDataset[time])
        tm ← list()
        for each line in SplittedDataset[time] do
            if time ≤ 12 : 00 and time > 05 : 00 then
                tm ← 0
            else if time ≤ 17 : 00 and time > 12 : 00 then
                tm ← 1
            else if time ≤ 22 : 00 and time > 17 : 00 then
                tm ← 2
            else if time ≤ 05 : 00 or time > 22 : 00 then
                tm ← 3
            end if
        end forreturn tm
    end function
    function createSMSDate(SplittedDataset[date])
        d ← list()
        for each line in SplittedDataset[date] do
            d ← count(date)
        end forreturn d
    end function
    function createSMSAf(SplittedDataset[name])
        a ← list()
        for each line in SplittedDataset[name] do
            a ← count(name)
        end forreturn a
    end function
    function createSMSCnt(SplittedDataset[place])
        cc ← list()
        for each line in SplittedDataset[place] do
            if place = local then
                cc ← 2
            else if place = unknown then
                cc ← 1
            else if place = foreign then
                cc ← 0
            end if
        end forreturn cc
    end function
    function createSMSMob(SplittedDataset[mobility])
        mob ← list()
        for each line in SplittedDataset[mobility] do
            if mobility = mobile then
                mob ← 2
            else if place = unknown then
                mob ← 1
            else if place = fixedline then
                mob ← 0
            end if
        end forreturn mob
    end function
    function createSMSLn(SplittedDataset[length])
        len ← list()
        for each line in SplittedDataset[length] do
            len ← length
        end forreturn len
    end function
    function createNNInp(t, tm, d, a, cc, mob, len)
        join(t, tm, d, a, cc, mob, len) return SMSInputs
    end function
end procedure
```

closer to one are regarded as more suspicious. Despite the fact that the output can receive any number within the aforementioned interval, five indicative values (0.15, 0.25, 0.5, 0.75, 1) were chosen as thresholds for each suspiciousness category. Table 5 demonstrates the assignment and the respective linguistic values.

TABLE 5. Fuzzy suspiciousness values, adapted from [7].

Value	Suspiciousness Level
0.15	Very Low
0.25	Low
0.5	Medium
0.75	High
1	Very High

This approach is also adopted in the current paper and is the basis of the ground truth generation process. Tuple combinations result in one out of the five aforementioned values. However, from the three NN and NFS methods used, only the plain backpropagation perceptron and ANFIS can make proper use of this method. The pattern recognition backpropagation perceptron requires additional output editing, because its format is based on binary states. In this perspective, five outputs are generated instead of one and one of them receives 1 as a value, whereas the rest of them remain 0s. Table 6 shows the output transformation for the pattern recognition backpropagation perceptron. In other words, the ground truth output template for the pattern recognition backpropagation perceptron is a 5×5 square diagonal matrix.

TABLE 6. Output transformations for the pattern recognition perceptron.

Value	Class 1	Class 2	Class 3	Class 4	Class 5
0.15	1	0	0	0	0
0.25	0	1	0	0	0
0.5	0	0	1	0	0
0.75	0	0	0	1	0
1	0	0	0	0	1

Subsection III-A provides a qualitative overview of the device usage MO for cyberbullying and low-level drug dealing. This information is rather useful for some first degree inferencing, but it cannot be precise enough without the appropriate numerical boundaries. These thresholds can be calculated after taking into consideration the CDA dataset from Subsection III-B, which includes mobile device usage for period of six months. This way, it is easier to define which variable ranges are considered high, medium or low. Each variable present in Equations 2 and 3 of Subsection III-C receives a specific value or interval of values and their combination can be translated into a statement, which is then assigned to a degree of suspiciousness. For example, a highly suspicious call for cyberbullying (Suspiciousness == 1) is performed at night, is missed or has a small duration, the bully's appearance frequency is relatively high, is performed by a mobile device and belongs to a local number. On the contrary, an innocent call (Suspiciousness == 0.15) is performed in the morning, has a very high or a very low duration, belongs to a fixed local line or a mobile line abroad.

The authors faced two main challenges during the ground truth generation phase. The first challenge was related to the manual labeling of the results, which was a rather time-consuming procedure, but it ensured their originality. However, in a future phase, this particular procedure can be replaced by a similar ground truth generation algorithm. The second challenge concerned the lack of suspicious patterns. During the manual labeling, there were no patterns that were classified as 1, i.e. the top suspiciousness scale. As a result, the authors had to generate a random number of suspicious patterns per dataset, based on the characteristics that classified them into the specific category and add them to the initial datasets in a randomized manner. Once the ground truth generation phase is complete, the preparation phase is concluded as well and the data are ready to be processed by the NN perceptrons and ANFIS.

TABLE 7. CDA GT pattern distribution.

Use Case Cat.	Cyberbullying Calls	Cyberbullying SMS	Drug Dealing Calls	Drug Dealing SMS
0.15	2,268	4,701	1,846	7,010
0.25	156	3,532	623	4,029
0.5	59	3,374	18	351
0.75	61	80	17	74
1	55	302	95	525
Total	2,591	11,989	2,591	11,989

E. NEURAL NETWORKS AND ANFIS CONFIGURATION

Three different neural and neuro-fuzzy network types, a plain backpropagation perceptron, a back-propagation pattern recognition perceptron and ANFIS are used for training and testing purposes. For every use case, seventy percent of the calls and SMS datasets is used for training, whereas fifteen percent is used for testing and the remaining fifteen for validation.

The follow-up procedure after the dataset splitting is equal for the plain and pattern recognition back-propagation perceptrons and different for ANFIS. The respective configuration settings will be presented in the following subsections.

1) PLAIN AND PATTERN RECOGNITION BACKPROPAGATION PERCEPTRON CONFIGURATION

The plain and pattern recognition perceptrons present in the current paper have a similar architecture. They consist of three layers, namely input, hidden and output. The input layer comprises seven inputs, as many as the input variables, whereas the output layer consists of one output for the plain backpropagation perceptron and five for the pattern recognition backpropagation. The architecture is depicted in Fig. 2. The decision-making procedure for the hidden layer is more complicated and is analytically discussed in the following paragraphs.

A highly-disputed claim that applies to the cases of the plain and pattern recognition backpropagation perceptrons is the number of hidden layers and nodes that are going to be used. Concerning the hidden layers issue, more than one layers are used in high complexity problems with a big number of inputs, such as image recognition or if the process

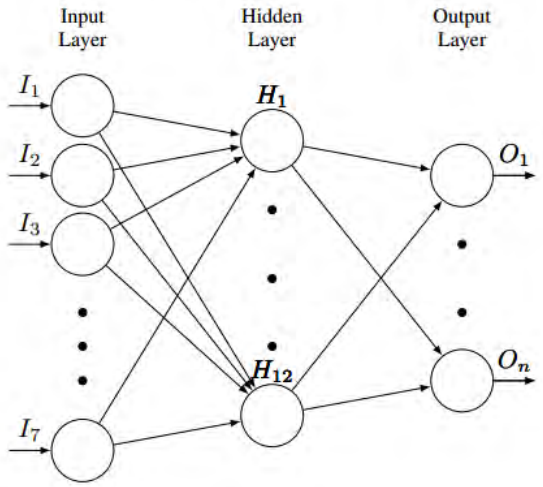


FIGURE 2. Generic plain and pattern recognition perceptron architecture.

being modeled is separable into multiple stages [23], whereas problems with a lower number of inputs, such as the one the current paper is trying to solve perform equivalently well with one hidden layer. On the contrary, there are many different arguments related to the number of hidden nodes scattered along the corresponding literature.

One of the approaches by Frontline Solvers [23] indicates that there should be an upper bound N_{max} to the number of hidden neurons per layer, which is given by the ratio of the total number of instances in the training dataset N_s divided by the sum of the number of inputs N_i and outputs N_o, multiplied by an arbitrary scaling factor α that receives values from 2 to 10. More details are provided in Equation 4.

$$N_{max} = \frac{N_s}{\alpha * (N_o + N_i)} \tag{4}$$

Other bibliographical sources are more precise in terms of defining the exact number of hidden nodes in a perceptron's hidden layer, but the diversity of the approaches is relatively big. Some of the referenced claims, broadly known as *rule-of-thumb methods* can be found below:

- "The number of hidden neurons should be between the size of the input layer and the size of the output layer" [31].
- "The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer" [51].
- "The number of hidden neurons should be less than twice the size of the input layer" [26].

However, many opinions conclude to the fact that there is no perfect rule to define the optimal number of hidden nodes and proceed to the adoption of trial-and-error methods starting from the lowest possible number of nodes and gradually increasing it until the lowest error rate is achieved [71]. After that point, the error increases anew. Moreover, difference in the performance between the training, validation and testing results also increases. While the perceptron achieves an excellent training performance rate, the error values in the testing or validation datasets are significantly higher (overfitting). Gethsiyal Augasta and Kathirvalavakumar [24] and Heaton [31] adopt a pruning approach, following the inverse procedure, i.e. beginning with a relatively high number of neurons and gradually reducing it.

$$MSE = \frac{1}{m}\sum_{p=1}^{m} e_p^2 = \frac{1}{m}\sum_{p=1}^{m}(t_p - o_p)^2 \tag{5}$$

In the current paper, the authors began experimenting in both the Calls and SMS datasets with the lowest number of hidden neurons and measured the performance rate for each step. *Mean Square Error (MSE)*, the mean value of the squared error e_p^2 for all the patterns of a dataset p is the difference between the expected and the actual outputs the perceptron produces and constitutes its performance indicator. A formal depiction of the MSE is shown in Equation 5, where m is the total number of patterns, t the vector corresponding to the target values and o the vector indicating the actual values the perceptron produced.

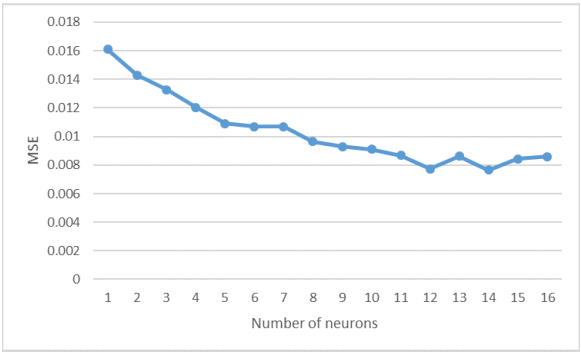


FIGURE 3. Average MSE per number of neurons - BP calls.

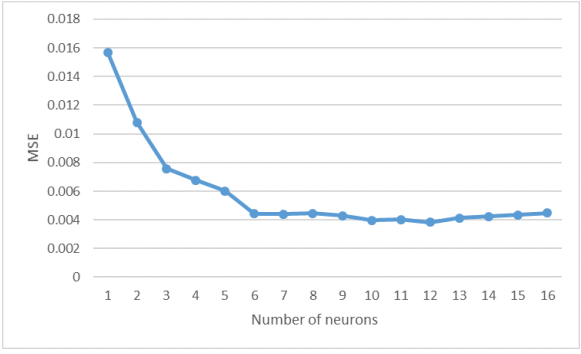


FIGURE 4. Average MSE per number of neurons - BP SMS.

Once the measurement phase was concluded, the decline in the MSE values was observed and the point where overfitting effects started appearing was encountered. Experiments were carried out for both plain backpropagation and pattern recognition perceptrons for every use case, data type and training algorithm. However, since the results were equal for all the different setups, the ones presented in the manuscript represent the whole picture. Fig. 3 and Fig. 4 show the

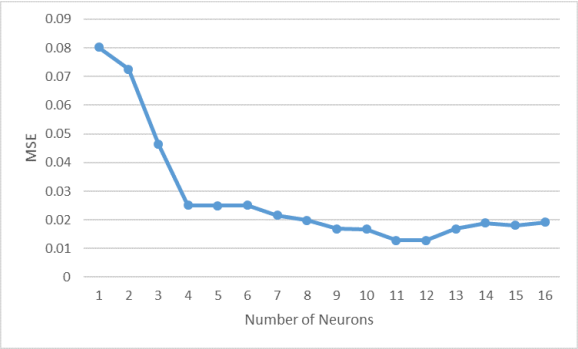


FIGURE 5. Average MSE per number of neurons - BPN calls.

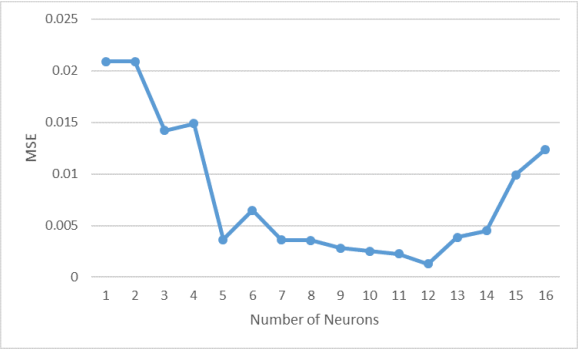


FIGURE 6. Average MSE per number of neurons - BPN SMS.

average plain backpropagation perceptron MSE values per number of neurons of the Calls and SMS Datasets. In Fig. 3, a considerable increase in the MSE values is observed when 13 neurons are used, signifying that after the specific point overfitting effects start to take place. Fig. 5 and Fig. 6 depict the average backpropagation pattern recognition perceptron MSE values per number of neurons of the Calls and SMS Datasets. In Fig. 5, when 11 and 12 neurons are used, the MSE levels are the lowest, fact that pinpoints that the optimal number of neurons is encountered in the aforementioned two values. Finally, the significant MSE decrease for the 5 neurons in Fig. 6 shows that satisfactory results can be produced at this stage. However, the lowest value is still observed when 12 neurons are used. Both the experimental setups ran the Levenberg-Marquardt backpropagation algorithm. Despite some increase exceptions, all the Figures indicate that the MSE rate decreases steeply after the second to third neuron addition and then decrease gradually until the twelfth neuron, where the lowest value is observed. From the thirteenth neuron and above, all the rates increase anew.

As a result of the aforementioned procedure, the plain and pattern recognition backpropagation perceptrons will carry twelve nodes in their hidden layer. However, before proceeding to the experimental and comparison phase, an introduction to the algorithms that were tested needs to be performed.

a: LEVENBERG-MARQUARDT BACKPROPAGATION

The Levenberg-Marquardt backpropagation algorithm is a hybrid of the steepest descent method and the Gauss-Newton algorithm. Its main goal is the optimized minimization of the global error function $E(x, w)$ depicted in Equation 6, where x and w are the input and weight vectors, m and n are the total numbers of patterns p and outputs y and $e^2_{p,y}$ is the squared value of the error, as already described in Subsection III-E1.

$$E(x, w) = \frac{1}{2}\sum_{p=1}^{m}\sum_{y=1}^{n} e^2_{p,y} = \frac{1}{2}\sum_{p=1}^{m}\sum_{y=1}^{n}(t_{p,y} - o_{p,y})^2 \quad (6)$$

The steepest descent method utilizes the first-order derivative of the global error function, so as to determine the minima within the error space, a characteristic that renders it "optimal for areas with complex curvature" [69]. When the aforementioned curvature allows for quadratic approximation, the respective error minimization algorithm selected is Gauss-Newton. In the Gauss-Newton algorithm, the weights vector is represented as a set of linearly independent gradient functions that are all set to zero for the estimation of the global error minima. Contrary to the steepest descent method, the second-order derivatives of the global error function, known as Hessian matrix H are calculated. In order to avoid complications caused by the calculation complexity of H driven by the second-order derivatives, the Jacobian matrix J is introduced instead, since, for the specific circumstances, H can be approximated as shown in Equation 7 [11].

$$H \approx J^T J \quad (7)$$

The combination coefficient μ, a positive value multiplied by the identity matrix I is added to Equation 7, so as to ensure that H remains always invertible. As a result, the Levenberg-Marquardt algorithm uses the following Equation.

$$H \approx J^T J + \mu I \quad (8)$$

The weights of a perceptron trained using the Levenberg-Marquardt algorithm are calculated by the following Equation, where w_ℓ and e_ℓ are the weight and error value of the ℓ-th node.

$$w_{\ell+1} = w_\ell - (J_\ell^T J_\ell + \mu_\ell I)^{-1} J_\ell e_\ell \quad (9)$$

When μ approaches to zero, the Gauss–Newton algorithm is used; when μ obtains a large value, the training algorithm swaps to the steepest descent method.

b: BAYESIAN REGULARIZATION

The Bayesian backpropagation algorithm is claimed to enhance the protection mechanism against perceptrons' overfitting and overtraining issues [64]. It "combines the conventional sum of the least squares error function with an additional term called *regularization*" [11]. This term, when added to the sum squared error equation, prevents the function from getting trapped into local minima [12] and the following cost function $S(w)$ is created, where α and β are the hyperparameters, E_p is the sum of squared errors

and E_w "is the penalty term, which penalizes large values of the weights" [11], with n being the maximum number of weights.

$$S(w) = \beta E_p + \alpha E_w = \beta \sum_{p=1}^{n} (t_p - o_p)^2 + \alpha \sum_{q=1}^{n} w_i^2 \quad (10)$$

The perceptron weights are regarded as random variables within the context of a Bayesian network [22]. As a result, the Bayes' theorem can be applied for the presentation of their density function.

$$P(w|X, \alpha, \beta, N) = \frac{P(X|w, \beta, N)P(w|\alpha, N)}{P(X|\alpha, \beta, N)} \quad (11)$$

In Equation 11, X is the input data vector, w refers to the perceptron weights' vector, while N is the perceptron model utilized.

c: BFGS QUASI-NEWTON BACKPROPAGATION

The *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* algorithm belongs to the quasi-Newton family of algorithms, which, contrary to Newton algorithms do not demand the calculation of a Hessian matrix due to potential lack of positive definite results or plain inefficiency [70]. They instead use a "symmetric positive definite approximation matrix A_n based on a rank-two correlation method" [25]. The inexact search scheme utilized "improves the computational scheme and allows the algorithm to have a global convergence property" [21]. The first step of the configuration includes the definition of a search direction S_n, where g_n is the gradient vector for each iteration.

$$s_n = -A_n^{-1} g_n \quad (12)$$

Once the direction is encountered, the search continues alongside so as to discover a step length σ_n that satisfies the following criterion.

$$f(w_n + \sigma_n s_n) = \min_{\sigma \geq 0}(f(w_n + \sigma_n s_n)) \quad (13)$$

Under the current circumstances, the weight vector w_n for the following step is formed as shown in Equation 14.

$$w_{n+1} = w_n + \sigma_n s_n \quad (14)$$

Afterwards, A_n gets updated to A_{n+1} with the rank-two correction provided below, where $\beta_n = w_{n+1} - w_n$ and $\gamma_n = g_{n+1} - g_n$.

$$A_{n+1} = A_n + \frac{\gamma_n \gamma_n^T}{\gamma_n^T \beta_n} - \frac{A_n \beta_n A_n \beta_n^T}{\beta_n^T A_n \beta_n} \quad (15)$$

The iteration comes to an end when a provided value λ is marginally greater or equal to the gradient of the objective function.

$$\sqrt{g_n^T g_n} \leq \lambda \quad (16)$$

d: ONE-STEP SECANT BACKPROPAGATION

The *One-Step Secant (OSS)* backpropagation algorithm was created as a means of "bridging the gap between the quasi-Newton and the Conjugate Gradient families of algorithms" [38]. In order to avoid storage issues, it omits storing the entire Hessian matrix and another approach is adopted instead. OSS is actually a memory-less BFGS variation. Instead of requiring an $O(N^2)$[20] amount of calculations order, its storage needs are reduced to $O(N)$. This is achieved "by obtaining the positive definite secant update for the inverse matrix A_{n+1}^{-1}" [9] from Equation 16.

$$A_{n+1}^{-1} = A_n^{-1} + \frac{(\beta_n - A_n^{-1}\gamma_n)\beta_n^T + \beta_n(\beta_n - A_n^{-1}\gamma_n^T)}{\gamma_n{}^T \beta_n} - \frac{< \beta_n - A_n^{-1}, \gamma_n > \beta_n^T \beta_n}{(\gamma_n^T \beta_n)^2} \quad (17)$$

Once each iteration ends, it is automatically inferred that the Hessian matrix of the previous step was the identity matrix I [59]. If Equation 17 is multiplied by the error gradient $g_n = \nabla(E(x, w))$, the next search direction (s_w) will be:

$$s_w = -g_n + C_n\beta_n + D_n\gamma_n = -g_n + [(-1 - \frac{\gamma_n^T \gamma_n}{\beta_n^T \gamma_n})\frac{\beta_n^T g_n}{\beta_n^T \gamma_n} + \frac{\gamma_n^T g_n}{\beta_n^T \gamma_n}]\beta_n + (\frac{\beta_n^T g_n}{\beta_n^T \gamma_n})\gamma_n \quad (18)$$

e: RESILIENT BACKPROPAGATION

The main scope of Resilient Backpropagation is to diminish the "adverse effects magnitudes of partial derivatives" [14] may cause to the weight update process, and consequently to the minimization of the error function. Thus, only the gradient signs are taken into consideration. The process is depicted in Equation 19.

$$w_n - w_{n-1} = -sign(g_{n-1})\Delta_n \quad (19)$$

The same initial Δ_0 is assigned to every update value. If the product of the current and the previous step is a positive number, i.e. if the vectors have the same direction, then a value η^+, greater than 1, is multiplied by the update value. Otherwise, the product of η^-, a negative value less than 1, and the update value is calculated. The aforementioned relationship is depicted in Equation 20.

$$\Delta_n = \begin{cases} \eta^+ \Delta n - 1, & g_{n-1} g_n > 0 \\ \eta^- \Delta n - 1, & g_{n-1} g_n < 0 \end{cases} \quad (20)$$

f: CONJUGATE GRADIENT BACKPROPAGATION FAMILY

Contrary to the steepest descent backpropagation algorithms, which use the negative gradient direction so as to achieve a faster reduction of the error function rates, the conjugate gradient family of backpropagation algorithms shares another common principle. Their basic aim is the production of more rapid convergence rates, so search is performed in the span of conjugate directions [58] and "the step size is adjusted at each iteration" [38].

The very first iteration in all the members of the Conjugate Gradient family of algorithms sets the search direction towards the negative gradient. Equation 21 shows the particular relationship, with s_0 indicating the search gradient and g_0 the initial gradient.

$$s_0 = -g_0 \tag{21}$$

Afterwards, linear search is employed so as to encounter the most appropriate moving distance. Equation 22 shows the status of the next weight w_{n+1}, which is the sum of the actual weight w_n and the product of the learning rate λ and the current search gradient s_n.

$$w_{n+1} = w_n + \lambda_n s_n \tag{22}$$

A prerequisite for the selection of the next search direction is to remain conjugate to the previous ones. The search direction is then defined anew as the combination of ''the new steepest descent direction with the previous search direction'' [38]. More details are provided in Equation 23, where s_{n-1} is the last search direction and β_n is a constant value that may vary from algorithm to algorithm.

$$s_n = -g_n + \beta_n s_{n-1} \tag{23}$$

Fletcher-Reeves Updates: For the Fletcher-Reeves Updates variation, the search direction is defined by Equation 23 and the aforementioned constant β_n is defined as ratio of the squared norm of the current gradient and the squared norm of the last one.

$$\beta_n = \frac{||g_n||^2}{||g_{n-1}||^2} \tag{24}$$

Polak-Ribière Updates: Similarly to the Fletcher Reeves variation, the Polak-Ribière Updates use Equation 23 for the calculation of the search direction. However, β_n is defined as the division of ''the inner product of the previous change in the gradient with the current gradient divided by the square of the previous gradient'' [38].

$$\beta_n = \frac{\Delta g_n^T g_n}{g_{n-1}^T g_{n-1}} \tag{25}$$

This method, as a four-vector algorithm, requires a bit higher amount of storage resources that the three-vector Fletcher-Reeves equivalent.

Powell-Beale Restarts: All members of the Conjugate Gradient family require a certain, periodic amount of resets to the original negative gradient value. Each restart takes place as soon as the number of iterations performed reaches the total number of the perceptron's biases and weights. However, restarts are not exclusively obligatory, but can rather occur at any given moment during the training phase, so as to improve its overall performance. The Powell-Beale restarts take place whenever the absolute value of the product $g_n^T g_n$ is greater than or equal of 0.2 times the squared norm of the current gradient. In terms of storage, the six-vector Powell-Beale Restarts require more space than the two aforementioned variations.

$$|g_{n-1}^T g_n| \geq 0.2||g_n||^2 \tag{26}$$

Scaled Conjugate Gradient: Contrary to the rest Conjugate Gradient algorithms, the Scaled Conjugate Gradient variation avoids performing searches per-line iteration and ''uses a step-sized scaling mechanism instead'' [58]. Due to computational complexity, the Hessian matrix H is approximated as shown in Equation 27, where w_n is the n-th weight, p_n is the n-th search direction, E' is the error gradient and σ_n and λ_n are scaling factors predefined by the user [3].

$$H_n = \frac{E'(w_n + \sigma_n p_n) - E'(w_n)}{\sigma_n} + \lambda_n p_n, \quad \sigma \in (0, 10^{-4}), \quad \lambda \in (0, 10^{-6}) \tag{27}$$

Additionally, the constant β_n is defined as follows:

$$\beta_n = \frac{|g_{n+1}|^2 - g_{n+1}^T g_n}{g_n^T g_n} \tag{28}$$

g: GRADIENT DESCENT BACKPROPAGATION FAMILY

The Gradient Descent family of algorithms contains some of the simplest backpropagation implementations. In this paper, the plain Gradient Descent and its variations (Gradient Descent with Momentum and Gradient Descent with Momentum and Adaptive Learning Rate) are examined.

Gradient Descent: Gradient Descent constitutes one of the plainest forms of error minimization techniques. The weight update is introduced as the product of the learning rate λ and the negative error gradient.

$$\Delta w(n) = -\lambda \nabla E(w) \tag{29}$$

Deciding on an appropriate learning rate is a rather difficult task and depends highly on the shape of the error function. Moreover, too high or too low values may lead to poor performance. One of the most known issues Gradient Descent algorithm face is the local minima trap, that does not allow for further (global) minimization of the error function.

Gradient Descent with Momentum: A simple variation of the Gradient Descent algorithm is the addition of a momentum term. In other words, ''the parameter μ scales the influence of the previous weight-step on the current one'' [53].

$$\Delta w(n) = -\lambda \nabla E(w) + \mu \Delta w(n-1) \tag{30}$$

Gradient Descent with Momentum and Adaptive Learning Rate: Momentum by itself is not enough to avoid complications deriving from rather poor choice of learning rate values. For this purpose, the algorithm is enhanced by the adaptive learning rate, which ''attempts to maintain the learning step size as large as possible while, keeping learning stable'' [46].

$$\Delta w(n) = -\lambda \mu \nabla E(w) + \mu \Delta w(n-1) \tag{31}$$

2) ANFIS

Adaptive Network-based Fuzzy Inference System (ANFIS) was first introduced by Jang [32] and combines the precision of Fuzzy Logic calculations with the adaptability of NNs. It eradicates the learning incapability of Fuzzy Systems, while it simultaneously makes use of the NN learning methods so as to efficiently tune fuzzy parameters, such as membership functions and their positioning. The learning algorithm present in ANFIS is inspired by the structure of a Takagi-Sugeno Fuzzy System. A plain form, comprising two inputs, one output and a base of two rules is adopted for demonstration purposes.

$$Rule1 : if\, x = A_1 and y = B_1, then f_1 = k_1 x + l_1 y + m_1$$
$$Rule2 : if\, x = A_2 and y = B_2, then f_2 = k_2 x + l_2 y + m_2$$

While x and y are the inputs, A_1, A_2, B_1 and B_2 are the membership functions corresponding to parts of the linguistic variables. The values k_1, k_2, l_1, l_2, m_1 and m_2 constitute linear parameters.

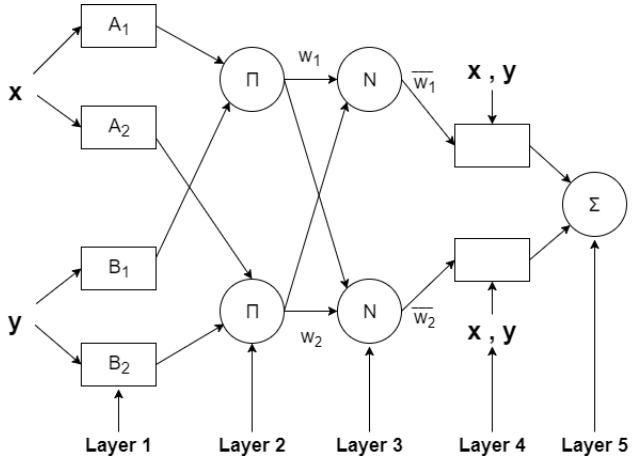


FIGURE 7. ANFIS, adapted from [32] and [63].

A simplified representation of ANFIS is provided in Fig. 7. The first layer comprises the system inputs. Contrary to the aforementioned perceptrons, each input does not correspond to the full range of a variable, but rather to separate partial fuzzy membership of it. From the previous claim, it can easily be inferred that the ANFIS input space is more complicated and fragmented than the one belonging to a plain or pattern recognition perceptron. The output produced by the first layer is provided by the following Equation, where μ is the membership function per input partition, usually Gaussian or Bell and $i \in Z_+^*$.

$$Ø_{1,i} = \begin{cases} \mu_{A_i}(x) \\ \mu_{B_i}(y) \end{cases} \tag{32}$$

The second layer consists of fixed Π nodes and corresponds to rule formulation by combining the appropriate membership values. Their output is calculated after the addition of a "T-norm operator to the existing membership value and represents the firing strength of each rule" [63]. Its format is shown in Equation 33.

$$Ø_{2,i} = w_i = \mu_{A_i}(x) * \mu_{B_i}(y) \tag{33}$$

The third layer also comprises non-adaptive N nodes and performs normalization of each rule's firing strength. This is achieved by dividing the current strength of the rule by the total strength of every rule encountered in the system.

$$Ø_{3,i} = \overline{w_i} = \frac{w_i}{\sum_{i=1}^{n} w_i} \tag{34}$$

The fourth layer produces node functions as outputs, the structure of which is provided below. The set of $\{k_i, l_i, m_i\}$ is known as a consequent parameters set.

$$Ø_{4,i} = \overline{w_i} f_i = \overline{w_i}(k_i x + l_i y + m_i) \tag{35}$$

Finally, the fifth layer is responsible for the global output calculation. The global output is defined as "the summation of all incoming signals" [32].

$$Ø_{5,i} = \sum_{i=1}^{n} \overline{w_i} f_i \tag{36}$$

In order to avoid the local minima trap issue deriving from the use of traditional backpropagation algorithms, such as Gradient Descent, a hybrid version, consisting of two opposite direction paths, was proposed. During the forward path, signals reach the fourth layer and a *Least Square Estimate (LSE)* algorithm is employed so as to determine the set of consequent parameters. Once this step is complete, the new data are deployed as inputs, and then the respective outputs are calculated and compared to the target outputs. "The consequent parameters remain in a steady state for the backward path" [63], as a reference point. The error resulting from the aforementioned comparison is forwarded anew to the first layer and Gradient Descent or other backpropagation algorithms are used for further optimization and final convergence.

Before proceeding to the utilization of the three different perceptrons, it is advisable to normalize the respective inputs and outputs. Normalization is the procedure of "rescaling the input and output variables independently by the minimum and range of the vector, to make all the elements lie in the set of [0,1]" [35]. This procedure can either take place manually, or automatically by the features of the Matlab NN Toolbox.

F. EVALUATION

One of the issues that the authors faced in this phase was the decision upon a common evaluation method. The plain backpropagation perceptron and ANFIS are by default regression models, whereas the backpropagation pattern recognition perceptron solves classification problems. However, the fact that the regression model outputs receive approximate values to the ones present in the ground truth set $S_G = \{0.15, 0.25, 0.5, 0.75, 1\}$ allow for a more detailed classification, in order to figure out if the produced outcomes match

the expected ones. Five machine learning algorithms, namely *k-Nearest Neighbor (kNN)*, *Support Vector Machine (SVM)*, *Random Forest Classification (RFC)*, *Naive Bayes (NB)* and *AdaBoost (Ada)* are employed in order to classify the perceptrons' test dataset outcomes in comparison to the ground truth targets and the algorithm with the best performance is selected as the final result. The sampling technique applied is 10-fold cross-validation.

The aforementioned procedure produces a 5×5 confusion matrix and classification metrics (Accuracy, Precision and Recall) are calculated for each category and then presented as an average score. More precisely, Accuracy refers to the ratio of the correctly classified patterns per category and the total number of patterns.

$$Accuracy = \left\langle \frac{TP_c + TN_c}{TP_c + FP_c + FN_c + TN_c} \right\rangle, \quad c \in S_G \tag{37}$$

Precision is the amount of *True Positive (TP)* patterns over the sum of TP and *False Positive (FP)* values.

$$Precision = \left\langle \frac{TP_c}{TP_c + FP_c} \right\rangle, \quad c \in S_G \tag{38}$$

Recall is the number that results from the division of the TPs with the total of TP and *False Negative (FN)* patterns.

$$Recall = \left\langle \frac{TP_c}{TP_c + FN_c} \right\rangle, \quad c \in S_G \tag{39}$$

Equations 37, 38 and 39 describe how each regression metric is calculated for every class c that belongs to the S_G set of suspiciousness values. The evaluation procedure is completed with the selection of the most efficient perceptron type.

G. TESTING ON UNKNOWN DATA

Once the perceptron with the best overall performance is identified, the following step is its test run on entirely unknown data. For that purpose, the authors performed a series of experiments on a Samsung Galaxy Ace 2 (GT-I8160) device, which was used by the first author for six consecutive months. The device was running the Android 4.4 version. *Android Data Acquisition and Examination Tool (ADAET)*, a Python script, was implemented so as to extract the appropriate databases, perform the pre-processing, invoke the Matlab scripts for the NN testing and calculate the equivalent metrics.

ADAET initially establishes an ADB connection between the target device and a workstation. After verifying that the device is rooted, the *mmssms.db* and *calls.db* databases are copied and saved at the workstation. However, the data in their raw form are not in the appropriate format to be processed by the NN. Afterwards, they are pre-processed by the algorithm presented in Subsection III-C. At a next step, ADAET invokes the Matlab scripts written for the plain backpropagation perceptron in Section III-E and lastly, the regression accuracy and other metrics are calculated. The aforementioned procedure is depicted in Fig. 8.

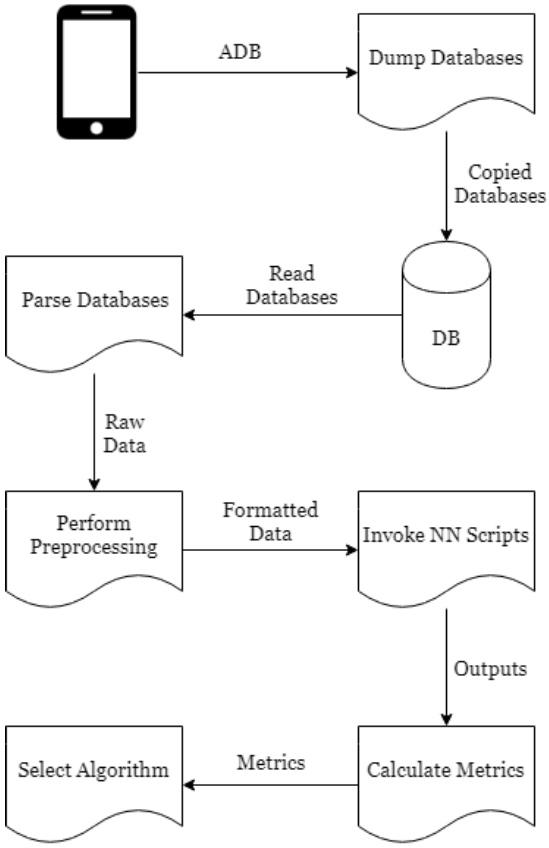


FIGURE 8. ADAET functionality, extended from [61].

Once the definition of the methodology is complete, the aforementioned steps are followed towards the results generation, that are presented analytically in the section that follows.

IV. RESULTS

This section presents the results that were generated after the conclusion of the experimental process. Initially, the results per use case and perceptron type for the CDA (training) dataset are provided. Afterwards, they are evaluated and the best performing perceptron and algorithm are selected. Once the aforementioned procedure is complete, ADAET acquires and processes calls and SMS data from the experimental mobile device and tests the performance metrics of the previously trained perceptron.

A. CDA DATASET RESULTS

The first step of the results presentation is associated to the performance evaluation of three different perceptron types (plain, backpropagation and ANFIS) for calls and SMS data, per corresponding use case. The next section presents the results for the Cyberbullying use case.

1) CYBERBULLYING

This section is divided in three parts; the first comprises the results for the plain backpropagation perceptron, the second for the pattern recognition backpropagation perceptron and the third for ANFIS.

TABLE 8. Plain backpropagation perceptron performance - Cyberbullying: Calls.

Algorithm	Perf.	Acc.	Prec.	Rec.
Levenberg-Marquardt	0.00179	92.4	77.5	74.4
Bayesian Regularization	0.000341	97.4	80.9	82.6
BFGS Quasi-Newton	0.00461	91.1	68.9	57.2
Scaled Conjugate Gradient	0.00398	91.1	75.4	72.7
One-Step Secant	0.00874	89.2	67.4	57.1
Resilient	0.00504	89.9	59.4	55.4
G. D.	0.0167	90.0	71.4	56.7
G. D. Momentum	0.0306	87.5	49.1	52.3
G. D. Momentum-Adaptive	0.0081	91.3	76.3	66.3
C. G. Powell-Beale	0.00427	90.8	66.3	59.6
C. G. Polak-Ribière	0.00392	91.4	76.6	49.7
C. G. Fletcher-Reeves	0.00428	89.9	69.7	63.7

TABLE 9. Plain backpropagation perceptron performance - Cyberbullying: SMS.

Algorithm	Perf.	Acc.	Prec.	Rec.
Levenberg-Marquardt	0.00277	92.4	87.7	71.4
Bayesian Regularization	0.00034	83.7	74.3	72.1
BFGS Quasi-Newton	0.00461	81.6	75.9	73.5
Scaled Conjugate Gradient	0.00398	78.5	74.3	71.4
One-Step Secant	0.00874	78.4	72.7	71.3
Resilient	0.00504	79.7	73.5	71.0
G. D.	0.0167	71.3	70.4	67.7
G. D. Momentum	0.0306	72.5	71.3	70.5
G. D. Momentum-Adaptive	0.0081	71.4	62.7	60.9
C. G. Powell-Beale	0.00427	73.5	64.0	61.7
C. G. Polak-Ribière	0.00392	78.3	70.0	67.6
C. G. Fletcher-Reeves	0.00428	79.6	77.0	73.3

a: PLAIN BACKPROPAGATION PERCEPTRON

For the calls dataset, the training algorithms showed almost excellent performance, with accuracy scores varying from 87.5 to 97.4%. The Levenberg-Marquardt and Bayesian Regularization algorithms outperform the rest, with the latter achieving the highest Accuracy score. In terms of Precision and Recall, only Bayesian Regularization scores over 80% and Levenberg-Marquardt follows with results in the mid-70s range. The rest of the algorithms have rather poor or unbalanced rates. More details can be found in Table 8, where the "Perf." abbreviation stands for the MSE rate per algorithm and "Acc.," "Prec" and "Rec." present the corresponding Accuracy, Precision and Recall values. Fig. 9 depicts the aforementioned metrics per algorithm.

For the SMS dataset, all the training algorithms scored within the 71.3-92.4% regression accuracy spectrum, that constitutes a fair to very good performance, but lower than the Calls dataset equivalents. Levenberg-Marquardt, Bayesian Regularization and BFGS Quasi-Newton backpropagation achieved the higher Accuracy scores (>80%), whereas the Gradient Descent family of algorithms showed the poorest performance. Levenberg-Marquardt backpropagation shows significantly higher Precision scores for each suspiciousness category. The Recall rates are slightly lower and all the algorithms perform equally. Table 9 shows the performance metrics for the SMS dataset,

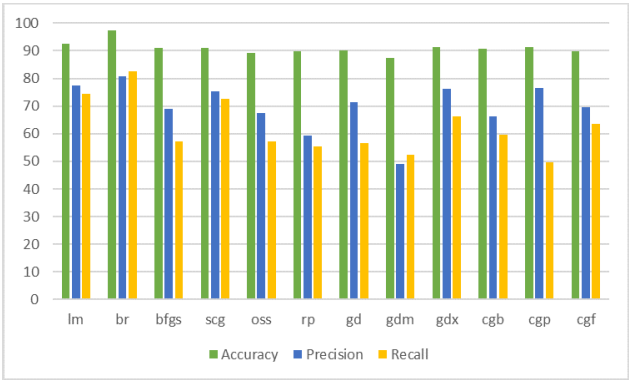


FIGURE 9. Performance histogram for the plain backpropagation perceptron - Cyberbullying: Calls.

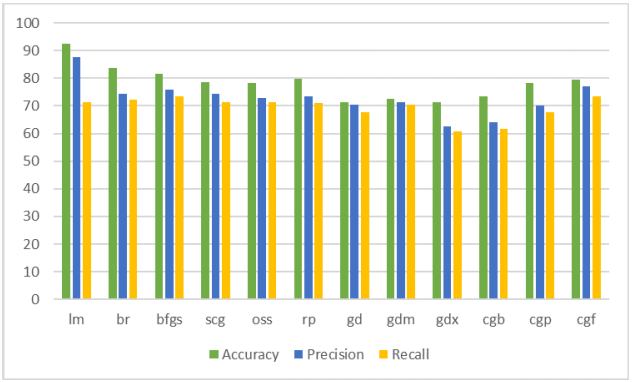


FIGURE 10. Performance histogram for the plain backpropagation perceptron - Cyberbullying: SMS.

b: BACKPROPAGATION PATTERN RECOGNITION NEURAL NETWORK

Performance for the Pattern Recognition perceptron is not as uniform as the Plain Backpropagation perceptron's. Accuracy for the Calls training, validation and testing datasets varies from a minimum 41.6% to a maximum 99.8%. However, only the validation and testing subsets are taken higher into consideration. Similarly to the results of the previous subsection, the Levenberg-Marquardt and Bayesian Regularization outperform the rest of the algorithms. Nevertheless, Levenberg-Marquardt shows a more balanced profile between training, testing and validation, whereas Bayesian Regularization, that by default lacks a validation dataset, presents a declining of almost six points. Once again, the Gradient Descent algorithms other than the variation with momentum and adaptive learning show the worst performance results. A more detailed overview is provided in Table 10.

The declining between the performance rates is smaller for the SMS dataset. The accuracy range is defined between 81.2% and 96.3%, with the Levenberg-Marquardt and Bayesian Regularization algorithms scoring the highest numbers anew. The difference between the rest of the algorithms is insignificant and only the simple Gradient Descent variation

TABLE 10. Backpropagation pattern recognition perceptron performance - Cyberbullying: Calls.

Algorithm	Perf.	Training	Validation	Testing
Levenberg-Marquardt	0.00998	95.4	93.2	94.9
Bayesian Regularization	0.000442	99.8	-	94.0
BFGS Quasi-Newton	0.0761	86.1	87.6	87.2
Scaled Conjugate Gradient	0.0690	89.4	87.6	87.2
One-Step Secant	0.0933	83.0	81.3	84.4
Resilient	0.0768	87.3	84.4	88.3
Gradient Descent (G.D.)	0.234	45.0	41.6	46.0
G. D. Momentum	0.229	54.9	57.9	56.6
G. D. Momentum-Adaptive	0.0953	83.0	84.4	83.5
C. G. Powell-Beale	0.05846	90.8	87.4	88.3
C. G. Polak-Ribière	0.0738	86.9	84.4	87.4
C. G. Fletcher-Reeves	0.0867	83.0	85.5	82.6

TABLE 11. Backpropagation pattern recognition perceptron performance - Cyberbullying: SMS.

Algorithm	Perf.	Training	Validation	Testing
Levenberg-Marquardt	0.0126	96.3	94.8	95.4
Bayesian Regularization	0.0166	95.6	-	92.6
BFGS Quasi-Newton	0.0478	90.1	91.5	90.6
Scaled Conjugate Gradient	0.0445	91.6	88.9	89.7
One-Step Secant	0.0454	91.7	92.3	88.0
Resilient	0.0347	92.7	94.0	88.9
G. D.	0.111	86.1	87.2	81.2
G. D. Momentum	0.145	84.8	88.0	89.7
G. D. Momentum-Adaptive	0.0412	91.9	91.5	92.3
C. G. Powell-Beale	0.0436	89.9	94.0	90.6
C. G. Polak-Ribière	0.0529	89.2	86.3	85.5
C. G. Fletcher-Reeves	0.0346	91.7	90.6	87.2

shows the lowest scores. Table 11 presents the respective results.

c: ANFIS

Due to the big number of inputs and corresponding linguistic variable subdivisions, it was impossible to create fuzzy systems manually (Type-1) by generating them from the data (Type-2). As a result, fuzzy clustering was the only available option in order to create the input space. Modifications in the squash factor and range of influence values resulted in different numbers of membership functions.

Despite the variations among the number of membership functions per instance, the difference between the Error and Accuracy rates do not surpass 2% for the Calls dataset. Moreover, despite the rather satisfactory average Accuracy percentages scored, the Precision and Recall rates are rather low. The amount of membership functions was between 18 and 30, whereas the version with the best overall performance in terms of Accuracy, Precision and Recall is the second column of Table 12.

Similar conclusions can be extracted from the SMS dataset, where the difference between the highest and the lowest Error and Accuracy values is not greater than 3.5%. The higger amount of membership functions generated was 57, whereas the lower was 18. The Accuracy scores are slightly higher than the ones achieved for the Calls dataset. However, the Precision and Recall metrics are significantly higher, but yet

TABLE 12. ANFIS performance - Cyberbullying: Calls.

Version	1	2	3	4	5
Range of Influence	0.5	0.25	0.75	0.45	0.45
Squash Factor	1.25	1.25	1.25	1.15	1.05
M.F.s	20	30	18	24	27
Error	0.0661	0.0768	0.067	0.0604	0.0549
Accuracy	90.9	91.3	91.4	89.9	90.1
Precision	43.8	51.5	42.3	48.1	30.2
Recall	30.4	37.0	23.9	28.3	28.3

TABLE 13. ANFIS performance - Cyberbullying: SMS.

Version	1	2	3	4	5
Range of Influence	0.5	0.25	0.75	0.4	0.4
Squash Factor	1.25	1.25	1.25	1.15	1.05
M.F.s	22	57	13	30	38
Error	0.075	0.097	0.085	0.072	0.068
Accuracy	90.6	91.8	88.6	91.9	92.0
Precision	58.1	61.5	56.1	62.4	62.5
Recall	59.1	62.7	57.4	63.3	63.3

not within the acceptable rates for a very good performance. More details about the ANFIS performance of the SMS dataset can be encountered in Table 13. The version with the best performance rates can be found in the fourth column of the aforementioned table.

The next subsection delves into the results generated by the three different perceptrons for the Drug Dealing use case and provides more information that will lead to the appropriate method selection.

2) DRUG DEALING

The structure of the current section follows the pattern of section IV-A1, where each part shows the performance evaluation results for each perceptron type.

a: PLAIN BACKPROPAGATION PERCEPTRON

The results concerning the Calls dataset of the Drug Dealing use case show rather high Accuracy rates within the 83.7-98.2% spectrum and are also accompanied by excellent Precision and Recall metrics of the 90s scale, at least for the best performing algorithms. Similarly to the Cyberbullying use case, Levenberg-Marquardt and Bayesian Regularization showed the best performance rates in all the metric categories. Conjugate Gradient with Fletcher-Reeves Updates and Resilient backpropagation followed with almost equivalently high Accuracy rates, but relatively lower Precision and Recall scores. The lowest performance score was achieved by the Gradient Descent and Gradient Descent with Momentum algorithms. More details about the performance of the algorithms are depicted in Table 14 and in Fig. 11.

Similar results were encountered in the SMS dataset, the Accuracy of which, however, covered a broader area of ranges, varying from 66.6% to 97.8%. Moreover, the Precision and Recall metrics were the highest out of all the datasets for the plain backpropagation perceptron experiments. This is

TABLE 14. Plain backpropagation perceptron performance - Drug Dealing: Calls.

Algorithm	Perf.	Acc.	Prec.	Rec.
Levenberg-Marquardt	0.0275	96.2	89.2	94.3
Bayesian Regularization	0.00230	98.2	96.6	96.6
BFGS Quasi-Newton	0.0111	94.6	84.5	68.2
Scaled Conjugate Gradient	0.00613	92.9	70.9	63.6
One-Step Secant	0.00798	92.0	73.1	77.3
Resilient	0.00875	95.7	78.1	93.2
G. D.	0.0373	89.0	57.3	53.4
G. D. Momentum	0.0440	83.7	49.4	45.5
G. D. Momentum-Adaptive	0.00974	95.6	79.8	89.8
C. G. Powell-Beale	0.00605	92.4	73.8	54.5
C. G. Polak-Ribière	0.0102	93.4	74.4	69.3
C. G. Fletcher-Reeves	0.00538	96.7	88.0	92.0

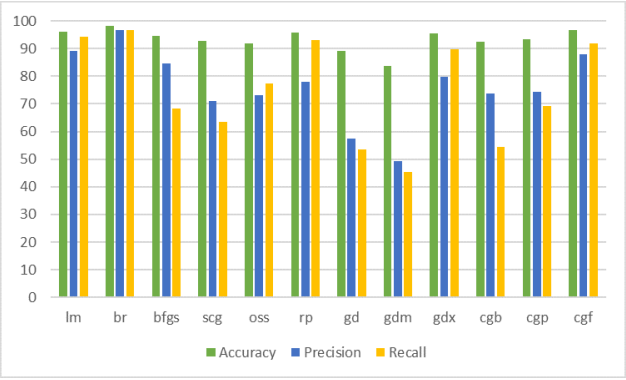


FIGURE 11. Performance histogram for the plain backpropagation perceptron - Drug Dealing: Calls.

TABLE 15. Plain backpropagation perceptron performance - Drug Dealing: SMS.

Algorithm	Perf.	Acc.	Prec.	Rec.
Levenberg-Marquardt	0.00123	96.5	95.3	96.2
Bayesian Regularization	0.000827	97.8	98.4	96.7
BFGS Quasi-Newton	0.0025	96.1	93.6	96.8
Scaled Conjugate Gradient	0.00217	95.6	92.6	96.6
One-Step Secant	0.00265	92.3	87.6	92.8
Resilient	0.00213	94.9	89.9	97.8
G. D.	0.0229	66.6	56.6	56.2
G. D. Momentum	0.0338	74.7	66.1	76.0
G. D. Momentum-Adaptive	0.00467	89.9	84.4	89.4
C. G. Powell-Beale	0.00255	93.3	89.4	92.9
C. G. Polak-Ribière	0.00221	96.6	96.0	95.9
C. G. Fletcher-Reeves	0.00191	97.0	96.8	96.0

the only dataset where the Levenberg-Marquardt algorithm did not have one of the two first places in performance, but achieved the third best scores after Bayesian Regularization and Conjugate Gradient with Polak-Ribière Updates. Gradient Descent and Gradient Descent with Momentum showed once again poor results. Table 15 and Fig. 12 analytically present the perceptron results for the SMS dataset.

b: BACKPROPAGATION PATTERN RECOGNITION PERCEPTRON

The current use case and dataset is an example of a non-successfully concluded experimental setup. All the

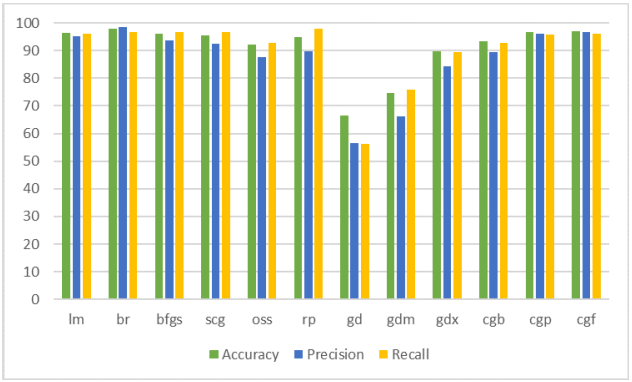


FIGURE 12. Performance histogram for the plain backpropagation perceptron - Drug Dealing: SMS.

algorithms failed to classify almost or more than half of the patterns of different suspiciousness for the Calls dataset. Both the training, validation and testing sessions did not provide an Accuracy score over 65%. The BFGS Quasi-Newton and Gradient Descent with Momentum and Adaptive Learning Rate performed slightly better than the rest of the algorithms, but the remaining members of the Gradient Descent family showed the worst results. More details about the scoring can be found in Table 16.

TABLE 16. Backpropagation pattern recognition perceptron performance - Drug Dealing: Calls.

Algorithm	Perf.	Training	Validation	Testing
Levenberg-Marquardt	0.111	56.2	47.4	51.1
Bayesian Regularization	0.116	54.7	-	57.1
BFGS Quasi-Newton	3.39	52.2	55.6	62.4
Scaled Conjugate Gradient	3.33	52.5	55.6	59.4
One-Step Secant	3.16	54.3	51.9	50.4
Resilient	3.17	54.6	52.6	53.4
G. D.	3.44	43.1	40.6	44.4
G. D. Momentum	3.34	41.7	41.4	47.4
G. D. Momentum-Adaptive	3.36	52.2	53.4	63.2
C. G. Powell-Beale	3.25	53.6	54.9	55.6
C. G. Polak-Ribière	3.27	53.5	57.9	53.4
C. G. Fletcher-Reeves	3.19	54.8	52.6	51.9

Contrary to the Calls dataset, the SMS dataset showed excellent Accuracy results that reached up to 99.7% for the training subset and 99.1% for the validation and testing equivalents. The Levenberg-Marquardt and Bayesian Regularization algorithms outperformed the rest and the lowest scores were marked for the Gradient Descent and Gradient Deswcent with Momentum algorithms. Table 17 presents the respective results.

3) ANFIS

Five different ANFIS versions were produced for each dataset. As far as the Calls dataset is concerned, the total amount of Membership Functions generated varied from 18 to 30. Despite the variety between their numbers, the Accuracy metrics were all similar and scored in the interval of 95.3 - 96.5%. Contrary to the cyberbullying use case, the

TABLE 17. Backpropagation pattern recognition perceptron performance - Drug Dealing: SMS.

Algorithm	Perf.	Training	Validation	Testing
Levenberg-Marquardt	0.000820	99.7	99.1	99.1
Bayesian Regularization	0.000899	99.7	-	98.3
BFGS Quasi-Newton	0.024	96.7	95.5	95.5
Scaled Conjugate Gradient	0.0146	97.3	97.0	96.8
One-Step Secant	0.0241	96.7	97.6	95.9
Resilient	0.0152	97.1	96.3	96.7
G. D.	0.158	72.8	75.7	73.7
G. D. Momentum	0.111	81.7	80.5	80.7
G. D. Momentum-Adaptive	0.0156	97.4	97.0	96.8
C. G. Powell-Beale	0.0161	96.9	97.8	95.7
C. G. Polak-Ribière	0.0146	97.3	97.4	96.8
C. G. Fletcher-Reeves	0.0174	97.2	96.1	97.6

Precision and Recall metrics were relatively high, over 80%. The best performance level was achieved by the fifth version, consisting of 27 membership functions. Analytical details are provided in Table 18.

TABLE 18. ANFIS performance - Drug Dealing: Calls.

Version	1	2	3	4	5
Range of Influence	0.5	0.25	0.75	0.45	0.25
Squash Factor	1.25	1.25	1.25	1.25	0.8
M.F.s	19	30	18	24	27
Error	0.098	0.0458	0.0525	0.0453	0.042
Accuracy	95.3	95.7	95.5	96.5	96.5
Precision	81.4	83.9	81.0	86.3	86.4
Recall	89.8	88.6	92.0	93.2	93.5

Similar, but borderline lower performance was noted for the SMS dataset. The number of Membership Functions per version varied from 15 to 51 and the Accuracy scores from 86.3% to 93.2%. As it can be observed in Table 19, despite the considerable increase in the number of membership functions between the best performing versions 2 and 5, the subsequent increase the performance score is very low. A fuzzy system with 51 membership functions is computationally slower than a version equipped with 26 and since the performance difference is rather low, the version with the lower number of membership functions can be selected as the most efficient in terms of performance and computational cost.

TABLE 19. ANFIS performance - Drug Dealing: SMS.

Version	1	2	3	4	5
Range of Influence	0.5	0.25	0.75	0.45	0.45
Squash Factor	1.25	1.25	1.25	1.15	1.05
M.F.s	15	51	10	22	26
Error	0.0665	0.0521	0.071	0.0622	0.0578
Accuracy	89.8	93.2	86.3	88.1	92.0
Precision	88.0	90.0	87.1	84.3	88.0
Recall	86.0	92.9	76.6	85.3	91.8

The aforementioned subsections showed that a decision upon the best approach for the identification of suspicious and non-suspicious patterns in mobile metadata is not a simplified procedure. However, there are some characteristics that clarify the selection procedure and are analytically presented in the following paragraphs.

B. BEST PERFORMANCE PERCEPTRON AND ALGORITHM SELECTION

Defining the most appropriate NN perceptron for the detection and rating of suspicious patterns is a rather complicated process, especially when the majority of the produced results are equivalently good. In such a case, the selection criteria are not limited to the success rates of each method, but focus on deeper levels of detail.

Generally, all the three perceptron types achieved a relatively high average performance rate, especially for their top variations. Accuracy rates over 80% were a common characteristic. Despite its satisfactory performance in three out of the four dataset and use case combination, the pattern recognition backpropagation perceptron performed significantly lower than expected for the Calls dataset of the Drug Dealing use case. On the contrary, the plain backpropagation perceptron and ANFIS did not face a similar issue. ANFIS showed an excellent performance profile for the Drug Dealing use case, but the Precision and Recall rates for the Cyberbullying use case were rather low. The plain backpropagation perceptron was the most stable method out of the three. Its results might not have reached the rates generated by the pattern recognition perceptron, but it was able to maintain a uniform average of results, especially for the best performing family of algorithms. As a result, the plain backpropagation perceptron is considered the preferred approach for solving the suspicious pattern detection problem from mobile forensic data.

Additionally, the observation and selection procedure also resulted in three noteworthy conclusions. Firstly, ANFIS is highly dependent on the amount of patterns under examination. Tables 12, 13, 18 and 19 indicate that the upper bound of the produced membership functions is significantly higher for the SMS than the Calls datasets. This can be justified by the fact that the amount of patterns in the SMS dataset is almost six times bigger than the Calls equivalent, as seen in Table 7. This observation though brings a scalability issue to the surface. As already mentioned in the previous section, ANFIS versions with many membership functions come at a high computational cost. Consequently, the ANFIS problem solving capability is finite and its performance versus efficiency ratio drops as the number of the patterns in the input space increases.

Secondly, the regression approach of the plain backpropagation perceptron is more efficient than the classification approach of the pattern recognition backpropagation network. The difference between the plain backpropagation and pattern recognition backpropagation perceptrons performance for the Calls dataset of the Drug Dealing use case is substantially considerable. While the former was able to detect most of the patterns correctly, the latter failed at the classification of a little less than 50%. This statement is not useful as a standalone assumption. However, if Table 7 is

taken into consideration, it is noticeable that the specific dataset has a less proportional pattern distribution than the remaining ones. Moreover, it has a smaller amount of total patterns when compared to the SMS datasets. The aforementioned results signify that the plain backpropagation perceptron is more capable of correctly detecting patterns with uneven distribution, fact that renders it more suitable as a tool for real-life circumstances.

Lastly, as far as the backpropagation algorithms are concerned, the Levenberg-Marquardt and Bayesian Regularization methods showed by far the best results. On the one hand, Bayesian Regularization achieved higher performance rates but showed a considerable amount of difference between training and testing datasets. On the other hand, the Levenberg-Marquardt algorithm showed moderately lower performance rates, but maintained the result uniformity. The difference between the aforementioned algorithms and the remaining ones was remarkably observable. Conjugate Gradient with Fletcher-Reeves Updates and Resilient backpropagation provided satisfactory results, while the Scaled Conjugate Gradient and BFGS Quasi-Newton backpropagation algorithms follow with vaguely noticeable performance declining. Two members of the Gradient Descent family, simple Gradient Descent and its momentum variation had the worst performance rates for all the experimental setups.

Once the appropriate approach is selected, the research procedure continues with testing the plain backpropagation perceptron on completely unknown data that are previously acquired from a mobile device. This scenario is closer to real circumstances and will test if the perceptron and its respective algorithms' efficiency is aligned with the actual test results.

C. TESTING ON UNKNOWN DATA

The plain backpropagation perceptron showed overall better performance rates compared to the rest of the employed techniques. This section presents the behavior of the previously trained perceptron with the CDA dataset patterns when entirely unknown data are used as inputs to the system. However, a limitation considering the pattern distribution needs to be taken into consideration beforehand.

Table 7 presents the occurrences of patterns, classified by their suspiciousness level, according to the ground truth generation. Since the device operated in real-life circumstances, the uniformity between the occurrences is significantly lower than the CDA dataset's. The pattern distribution has an effect on the calculation of the regression accuracy and the other metrics. As already mentioned in Section III-F, the metrics are calculated by using 10-fold cross validation. However, when the number of patterns per category is less than 10, the respective metrics are omitted because the number of actual patterns is lower than the folds and no effective comparison can take place. It is also expected that the actual device datasets do not contain patterns of the highest suspiciousness level.

Table 21 presents the results for the Calls dataset of the Cyberbullying use case. The table constitutes an interesting case, because all the algorithms perform at approximately the same level, despite the differences encountered during the experimental phase in Section IV. Since the Accuracy metrics do not show significant differences, the Precision and Recall results will be examined. The Levenberg-Marquardt, Bayesian Regularization and BFGS Quasi-Newton algorithms have the most balanced performance. Surprisingly enough, in the specific sample, Gradient Descent shows a very efficient profile as well.

TABLE 20. Samsung device GT pattern distribution.

Use Case Cat.	Cyberbullying Calls	Cyberbullying SMS	Drug Dealing Calls	Drug Dealing SMS
0.15	400	383	238	597
0.25	66	152	246	482
0.5	31	544	8	–
0.75	3	–	7	–
1	–	–	–	–
Total	500	1079	500	1079

TABLE 21. Samsung device backpropagation perceptron performance - Cyberbullying: Calls.

Algorithm	Perf.	Acc.	Prec.	Rec.
Levenberg-Marquardt	0.1023	89.0	79.7	79.2
Bayesian Regularization	5.4039	88.8	79.6	80.0
BFGS Quasi-Newton	0.1026	89.0	76.8	84.3
Scaled Conjugate Gradient	0.6159	89.4	74.6	77.9
One-Step Secant	0.0393	89.0	84.2	68.0
Resilient	0.1125	90.0	71.7	78.7
G. D.	0.0850	88.0	81.3	85.7
G. D. Momentum	0.0473	88.4	68.5	76.1
G. D. Momentum-Adaptive	0.3622	89.0	77.5	76.0
C. G. Powell-Beale	0.4134	90.0	75.1	77.2
C. G. Polak-Ribière	0.1254	90.8	77.6	74.5
C. G. Fletcher-Reeves	0.0328	88.0	75.1	73.3

The performance of the SMS dataset is depicted in Table 22. The Levenberg-Marquardt algorithms performs significantly better than the rest of the backpropagation methods, with Bayesian Regularization and BFGS Quasi-Newton following closely. Bayesian Regularization shows a significant difference in its performance, when compared to the training and testing experimental phase. The Gradient Descent family of algorithms shows the poorest performance once again.

The results deriving from the examination of Table 23 for the Calls dataset of the Drug Dealing use case constitute a performance surprise. Firstly, none of the algorithms does not have an Accuracy score over 90%. Secondly, other than the rather expected Levenberg-Marquardt best performance, Gradient Descent with Momentum shows the best results in the category, with its Recall levels reaching almost 94%. The rest of the results are also uniform, with small variations.

Finally, Table 23 shows the performance rates of the SMS dataset. The produced results are rather impressive, with almost excellent metrics. This happens partially because of the existence of only two patterns in the dataset space, as it can be inferred from Table 20. Almost all the algorithms, other than the Gradient Descent and its variation with Momentum performed equally well.

Fig. 13 is a performance diagram of all the backpropagation algorithms for each use case and dataset of the actual

TABLE 22. Samsung device backpropagation perceptron performance - Cyberbullying: SMS.

Algorithm	Perf.	Acc.	Prec.	Rec.
Levenberg-Marquardt	0.0203	92.4	94.6	85.4
Bayesian Regularization	0.0048	83.7	73.3	72.1
BFGS Quasi-Newton	0.0139	81.6	75.9	73.5
Scaled Conjugate Gradient	0.0153	78.5	74.2	71.3
One-Step Secant	0.0158	78.4	72.6	71.2
Resilient	0.0169	78.7	73.5	71.0
G. D.	0.0332	71.3	70.4	67.7
G. D. Momentum	0.0509	72.5	71.2	70.4
G. D. Momentum-Adaptive	0.0211	71.4	62.7	60.9
C. G. Powell-Beale	0.0176	73.5	64.0	61.7
C. G. Polak-Ribière	0.0164	78.3	70.0	67.6
C. G. Fletcher-Reeves	0.0142	79.6	76.6	75.7

TABLE 23. Samsung device backpropagation perceptron performance - Drug Dealing: Calls.

Algorithm	Perf.	Acc.	Prec.	Rec.
Levenberg-Marquardt	0.383	85.4	82.4	89.4
Bayesian Regularization	0.8458	83.0	81.7	87.4
BFGS Quasi-Newton	0.0408	83.8	82.9	86.6
Scaled Conjugate Gradient	0.0729	83.6	83.8	86.2
One-Step Secant	0.1988	88.2	88.9	87.8
Resilient	0.1528	79.2	78.5	82.9
G. D.	0.1718	79.2	89.6	80.9
G. D. Momentum	0.2832	87.6	84.3	93.9
G. D. Momentum-Adaptive	0.407	81.2	79.1	86.2
C. G. Powell-Beale	0.19	85.8	86.8	85.8
C. G. Polak-Ribière	0.0375	81.4	80.6	82.9
C. G. Fletcher-Reeves	0.0602	80.4	80.6	82.5

TABLE 24. Samsung device backpropagation perceptron performance - Drug Dealing: SMS.

Algorithm	Perf.	Acc.	Prec.	Rec.
Levenberg-Marquardt	5.084e-04	99.6	99.7	99.6
Bayesian Regularization	0.0116	99.9	99.8	99.8
BFGS Quasi-Newton	7.177e-04	99.9	99.8	99.8
Scaled Conjugate Gradient	1.432e-04	99.9	99.8	99.8
One-Step Secant	1.687e-04	99.8	99.7	99.6
Resilient	8.402e-04	93.2	94.9	89.6
G. D.	0.015	91.9	86.6	96.9
G. D. Momentum	0.0178	82,0	92.2	51.7
G. D. Momentum-Adaptive	0.0211	94.5	85.4	85.6
C. G. Powell-Beale	1.755e-04	99.7	99.6	99.4
C. G. Polak-Ribière	3.072e-04	99.9	99.8	99.8
C. G. Fletcher-Reeves	7.390e-04	99.5	99.5	99.0

device testing case. The last column of each diagram subsection is the average performance of every algorithm. The superiority of the Levenberg-Marquardt, Bayesian Regularization and BFGS Quasi-Newton backpropagation can be inferred directly from the diagram. In general, the algorithms did not show a different behavior between the two different datasets. Levenberg-Marquardt showed the most balanced behavior, whereas the performance differences for Bayesian Regularization were a bit more considerable. As a result, the most appropriate combination for examining a mobile device for suspicious patterns and classifying the total patterns in different categories is the use of plain backpropagation perceptrons, trained by either Levenber-Marquardt or Bayesian Regularization algorithms. The next section will provide more details on how the existing work can be expanded and further improved.

FIGURE 13. Average performance per training algorithm.

V. DISCUSSION

This work showed that suspicious pattern detection and automatic pattern labeling from mobile metadata can be successfully concluded with the use of backpropagation perceptrons based on regression calculation. Pattern Recognition perceptrons, that are based on strict classification are not equivalently efficient and can provide rather low quality results when dealing with uneven data distributions. Neurofuzzy networks, such as ANFIS show satisfactory results but their configuration is significantly complicated and are prone to scalability issues when dealing with big loads of data.

It was expected that the results between the CDA training phase and the tests performed on the Samsung device would have some certain amount of decline, because it was the first time that the perceptron was handling completely unknown data. However, the difference in the performance was not significant and the perceptron for all the test device use cases scored with a regression Accuracy approximately and over 80%.

Despite the satisfactory results, there are many aspects in the current work that can be further improved. First and foremost, if data deriving from actual criminal investigations constituted the input space as a whole, the detection would be more accurate and relevant to realistic conditions, other than simulations. Secondly, the datasets can be enhanced with more features per data type, which can be retrieved by forensic acquisition performed on the target device. Some of the options include but are not limited to contact association to the device phonebook, existence of contact images, ring-until-pickup duration, etcetera. Lastly, more data types,

such as social media logs, coordinates and network traffic can be also examined for suspicious patterns and the device can then be examined in an complete level.

VI. CONCLUSIONS

The current paper constitutes a completed work on a methodology concerning the detection of suspicious patterns for cyberbullying and drug dealing digital fingerprints in call and SMS data from mobile devices. Based on a series of formal practices on criminal profiling, it uses the aforementioned procedure so as to build a behavioral model of digital criminal MO. Contrary to its predecessors that usually do not proceed beyond the modeling phase, its innovative feature is the fact that the aforementioned digital MO are translated into quantitative inputs, the combinations of which result in the creation of various patterns, corresponding to different degrees of suspiciousness.

Once the suspiciousness levels per pattern distributions are defined, plain backpropagation perceptrons, pattern recognition backpropagation perceptrons and ANFIS are tested for the efficiency of their backpropagation algorithms on the correct approximation of the initial degrees of suspiciousness per pattern. Despite the fact that all of them showed a relatively high performance level, the plain backpropagation perceptron is considered the most appropriate solution because of its speed, ease of implementation and uniformity between the provided results.

Lastly, the performance of the plain backpropagation perceptron is tested anew with a set of previously unknown data to the system, with a small declining in the performance values, fact that serves as an additional verification factor. Thus, the proposed methodology is proven to be an efficient solution for evidence handling and suspicious behavior detection with promising expansion capabilities.

ACKNOWLEDGMENT

The authors thank the team of the CATRENE Project Mobi-Trust (CA208) for the fruitful discussions and feedback.

REFERENCES

[1] M. Al Fahdi, N. L. Clarke, and S. M. Furnell, "Challenges to digital forensics: A survey of researchers & practitioners attitudes and opinions," in *Proc. Inf. Secur. South Afr.*, Johannesburg, South Africa, Aug. 2013, pp. 1–8.

[2] J. M. Alonso, C. Castiello, and C. Mencar, "Interpretability of fuzzy systems: Current research trends and prospects," in *Springer Handbook of Computational Intelligence*. Berlin, Germany: Springer, 2015.

[3] O. Baghirli, "Comparison of Lavenberg-Marquardt, scaled conjugate gradient and Bayesian regularization backpropagation algorithms for multi-step ahead wind speed forecasting using multilayer perceptron feedforward neural network," M.S. thesis, Dept. Earth Sci., Uppsala Univ., Uppsala, Sweden, 2015.

[4] K. Barmpatsalou, T. Cruz, E. Monteiro, and P. Simoes, "From fuzziness to criminal investigation: An inference system for mobile forensics," in *Intrusion Detection and Prevention for Mobile Ecosystems*, G. Kambourakis, A. Shabtai, C. Kolias, and D. Damopoulos, Eds. Boca Raton, FL, USA: CRC Press, 2017.

[5] K. Barmpatsalou, T. Cruz, E. Monteiro, and P. Simoes. *What Does Your Drug Dealer Sends you? A Compilation of OCRextracted Data From the Instagram Account @shotta_texts_ldn*. Accessed: Nov. 17, 2017. [Online]. Available: https://data.world/tbr/dealersms

[6] K. Barmpatsalou, T. Cruz, E. Monteiro, and P. Simoes, "Current and future trends in mobile device forensics: A survey," *ACM Comput. Surv.*, vol. 51, no. 3, 2018, Art. no. 46.

[7] K. Barmpatsalou, T. Cruz, E. Monteiro, and P. Simoes, "Fuzzy system-based suspicious pattern detection in mobile forensic evidence," in *Proc. 9th Int. Conf. Digit. Forensics Cyber Crime (ICDF2C)*, P. Matoušek and M. Schmiedecker, Eds. Prague, Czech Republic: Springer, Oct. 2017.

[8] K. Barmpatsalou, D. Damopoulos, G. Kambourakis, and V. Katos, "A critical review of 7 years of mobile device forensics," *Digit. Invest.*, vol. 10, no. 4, pp. 323–349, 2013.

[9] R. Battiti, "First- and second-order methods for learning: Between steepest descent and Newton's method," *Neural Comput.*, vol. 4, no. 2, pp. 141–166, Mar. 1992.

[10] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. New York, NY, USA: Springer, 2005.

[11] D. T. Bui, B. Pradhan, O. Lofman, I. Revhaug, and O. B. Dick, "Landslide susceptibility assessment in the Hoa Binh province of Vietnam: A comparison of the Levenberg–Marquardt and Bayesian regularized neural networks," *Geomorphology*, vols. 171–172, pp. 12–29, Oct. 2012.

[12] F. Burden and D. Winkler, "Bayesian regularization of neural networks," in *Artificial Neural Networks: Methods and Applications*, D. J. Livingstone, ed. New York, NY, USA: Humana Press, 2008.

[13] E. Casey and B. Turnbull, "Digital evidence on mobile devices," in *Digital Evidence and Computer Crime*, 3rd ed, E. Casey, ed. New York, NY, USA: Academic, 2011, pp. 1–44.

[14] S. Chabaa, A. Zeroual, and J. Antari, "Identification and prediction of Internet traffic using artificial neural networks," *J. Intell. Learn. Syst. Appl.*, vol. 2, no. 3, pp. 147–155, 2010.

[15] Digital Corpora. *Digital Corpora: Producing the Digital Body*. Accessed: Dec. 16, 2017. [Online]. Available: http://digitalcorpora.org

[16] A. Edwards. Drug dealer used unemployment benefits to make 24,000 mobile phone calls to sell heroin and crack. Daily Mail, U.K., Accessed: Jan. 11, 2017. [Online]. Available: http://www.dailymail.co.uk/news/article-2260633/Drug-dealer-used-unemployment-benefits-make-24-000-mobile-phone-calls-sell-heroin-crack.html.

[17] M. Enache, M. Hulea, and T. S. Leţia, "Training neural networks for construction of informatics offender profile," in *Proc. IEEE Int. Conf. Automat., Qual. Testing, Robot. (AQTR)*, Cluj-Napoca, Romania, May 2010, pp. 1–6.

[18] S. Ferrari, K. C. Baumgartner, G. B. Palermo, R. Bruzzone, and M. Strano, "Network models of criminal behavior," *IEEE Control Syst.*, vol. 28, no. 4, pp. 65–77, Aug. 2008.

[19] J. T. Fish, L. S. Miller, M. C. Braswell, and E. W. Wallace, Jr., "The electronic crime scene," in *Crime Scene Investigation*, 3rd ed. Boston, MA, USA: Anderson Publishing, 2014.

[20] J. Fleetwood, "Keeping out of trouble: Female crack cocaine dealers in England," *Eur. J. Criminol.*, vol. 11, no. 1, pp. 91–109, 2014.

[21] R. Fletcher, *Practical Methods of Optimization*. Chichester, U.K.: Wiley, 1987.

[22] F. D. Foresee and M. T. Hagan, "Gauss-Newton approximation to Bayesian learning," in *Proc. IEEE Int. Conf. Neural Netw.*, Houston, TX, USA, Jun. 1997, pp. 1930–1935.

[23] Frontile Solvers. *Training an Artificial Neural Network*. Accessed: Jan. 17, 2017. [Online]. Available: https://www.solver.com/training-artificial-neural-network-intro

[24] M. G. Augasta and T. Kathirvalavakumar, "Pruning algorithms of neural networks—A comparative study," *Central Eur. J. Comput. Sci.*, vol. 3, no. 3, pp. 105–115, 2013.

[25] A. Ghosh and M. Chakraborty, "Hybrid optimized back propagation learning algorithm for multi-layer perceptron," *Int. J. Comput. Appl.*, vol. 60, no. 13, pp. 1–5, 2012.

[26] K. G. Sheela and S. N. Deepa, "Review on methods to fix number of hidden neurons in neural networks," *Math. Problems Eng.*, vol. 2013, May 2013, Art. no. 425740. [Online]. Available: https://www.hindawi.com/journals/mpe/2013/425740/cta/

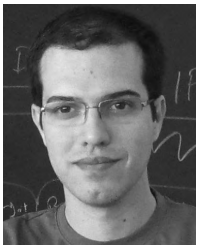
[27] M. Godwin. (2012). Brief discussion on inductive/deductive profiling. Godwin Trial & Forensic Consultancy. Accessed: Jan. 15, 2018. [Online]. Available: http://www.drmauricegodwin.com/inductiveprofiling.html

[28] A. K. Goodboy and M. M. Martin, "The personality profile of a cyberbully: Examining the dark triad," *Comput. Hum. Behav.*, vol. 49, pp. 1–4, Aug. 2015.

[29] A. Görzig and K. Ólafsson, "What makes a bully a cyberbully? Unravelling the characteristics of cyberbullies across twenty-five European Countries," *J. Children Media*, vol. 7, no. 1, pp. 9–27, 2013.

[30] C. Grajeda, F. Breitinger, and I. Baggili, "Availability of datasets for digital forensics—And what is missing," *Digit. Invest.*, vol. 22, pp. S94–S105, Aug. 2017.
[31] J. Heaton, *Introduction to Neural Networks for Java*, 2nd ed. Jefferson City, MO, USA: Heaton Res., 2008.
[32] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.
[33] J.-W. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim, "Andro-AutoPsy: Anti-malware system based on similarity matching of malware and malware creator-centric information," *Digit. Invest.*, vol. 14, pp. 17–35, Sep. 2015.
[34] Y. Jin, *Advanced Fuzzy Systems Design and Applications*. New York, NY, USA: Physica-Verlag, 2003.
[35] B. Iglewicz, "Robust scale estimators and confidence intervals for location," in *Understanding Robust and Exploratory Data Analysis*, D. C. Hoaglin, F. Mosteller, and J. W. Tukey, Eds. Hoboken, NJ, USA: Wiley, 1983.
[36] M. Islam and V. K. Verma, "Fuzzy logic based risk model for SMS threats in 3G systems," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 2, no. 2, pp. 1–5, 2012.
[37] D. Kasiaras, T. Zafeiropoulos, N. Clarke, and G. Kambourakis, "Android forensics: Correlation analysis," in *Proc. 9th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, London, U.K., Dec. 2014, pp. 157–162.
[38] J. Kuruvilla and K. Gunavathi, "Lung cancer classification using neural networks for CT images," *Comput. Methods Programs Biomed.*, vol. 113, no. 1, pp. 202–209, 2014.
[39] P. Lai, K.-P. Chow, X.-X. Fan, and V. Chan, "An empirical study profiling Internet pirates," in *Advances in Digital Forensics IX*, G. Peterson and S. Shenoi, Eds. Berlin, Germany: Springer, 2013.
[40] E. Lievens, "Bullying and sexting in social networks: Protecting minors from criminal acts or empowering minors to cope with risky behaviour?" *Int. J. Law Crime Justice*, vol. 42, no. 3, pp. 251–270, 2014.
[41] P. J. G. Lisboa, "Interpretability in machine learning—Principles and practice," in *Fuzzy Logic Applications*. Cham, Switzerland: Springer, 2013.
[42] J. Martin, "Lost on the silk road: Online drug distribution and the 'cryptomarket,"' *Criminol. Criminal Justice*, vol. 14, no. 3, pp. 351–367, 2014.
[43] T. May and M. Hough, "Drug markets and distribution systems," *Addiction Res. Theory*, vol. 12, no. 6, pp. 549–563, 2004.
[44] R. N. McEwen, "Tools of the trade: Drugs, law and mobile phones in Canada," *New Media Soc.*, vol. 13, no. 1, pp. 134–150, 2011.
[45] J. Mena, *Investigative Data Mining for Security and Criminal Detection*. Oxford, U.K.: Butterworth–Heinemann, 2003.
[46] S. Mohanty, M. K. Jha, A. Kumar, and K. P. Sudheer, "Artificial neural network modeling for groundwater level forecasting in a river Island of Eastern India," *Water Resource Manage.*, vol. 24, no. 9, pp. 1845–1865, 2010.
[47] M. Natarajan, "Understanding the structure of a large heroin distribution network: A quantitative analysis of qualitative data," *J. Quant. Criminol.*, vol. 22, no. 2, pp. 171–192, 2006.
[48] National Institute of Standards and Technology. (2016). *The CFReDS Project*. Accessed: Dec. 16, 2017. [Online]. Available: https://www.cfreds.nist.gov/
[49] C. Ntantogian, D. Apostolopoulos, G. Marinakis, and C. Xenakis, "Evaluating the privacy of Android mobile applications under forensic analysis," *Comput. Secur.*, vol. 42, pp. 66–76, May 2014.
[50] Openet. *Survey: 41% of Teens Experience Cyber-Bullying on Their Cellies*. Accessed: Dec. 15, 2017. [Online]. Available: https://www.openet.com/about-us/press-room/news/survey-41-teens-experience-cyber-bullying-their-cellies
[51] G. Panchal, A. Ganatra, Y. P. Kosta, and D. Panchal, "Behaviour analysis of multilayer perceptronswith multiple hidden neurons and hidden layers," *Int. J. Comput. Theory Eng.*, vol. 3, no. 2, pp. 332–337, 2011.
[52] D. Quick and K.-K. R. Choo, "Pervasive social networking forensics: Intelligence and evidence from mobile device extracts," *J. Netw. Comput. Appl.*, vol. 86, pp. 24–33, May 2017.
[53] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons—From backpropagation to adaptive learning algorithms," *Comput. Standards Interfaces*, vol. 16, no. 3, pp. 265–278, 1994.
[54] A. J. Roberto, J. Eden, M. W. Savage, L. Ramos-Salazar, and D. M. Deiss, "Prevalence and predictors of cyberbullying perpetration by high school seniors," *Commun. Quart.*, vol. 62, no. 1, pp. 97–114, 2014.
[55] M. Rogers, "The role of criminal profiling in the computer forensics process," *Comput. Secur.*, vol. 22, no. 4, pp. 292–298, 2003.
[56] M. K. Rogers, "Psychological profiling as an investigative tool for digital forensics," in *Digital Forensics: Threatscape and Best Practices*. Amsterdam, The Netherlands: Elsevier, 2016.
[57] D. T. Sacco, R. Argudin, J. Maguire, and K. Tallon, "Sexting: Youth practices and legal implications," Cyberlaw Clinic, Harvard Law School, Cambridge, MA, USA, Tech. Rep. 2010-8, 2010.
[58] R. Shaharin, U. K. Prodhan, and M. Rahman, "Performance study of TDNN training algorithm for speech recognition," *Int. J. Adv. Res. Comput. Sci. Technol.*, vol. 2, no. 4, pp. 90–95, 2014.
[59] T. N. Singh, A. R. Gupta, and R. Sain, "A comparative analysis of cognitive systems for the prediction of drillability of rocks and wear factor," *Geotech. Geological Eng.*, vol. 24, pp. 299–312, Apr. 2006.
[60] P. K. Smith, J. Mahdavi, M. Carvalho, S. Fisher, S. Russell, and N. Tippett, "Cyberbullying: Its nature and impact in secondary school pupils," *J. Child Psychol. Psychiatry*, vol. 49, no. 4, pp. 376–385, 2008.
[61] M. Spreitzenbarth and J. Uhrmann, *Mastering Python Forensics*. Birmingham, U.K.: Packt, 2015.
[62] M. Strano, "A neural network applied to criminal psychological profiling: An Italian initiative," *Int. J. Offender Therapy Comparative Criminol.*, vol. 48, no. 4, pp. 495–503, 2004.
[63] W. Suparta and K. M. Alhasa, "Adaptive neuro-fuzzy interference system," in *Modeling of Tropospheric Delays Using ANFIS*. Cham, Switzerland: Springer, 2016.
[64] J. L. Ticknor, "A Bayesian regularized artificial neural network for stock market forecasting," *Expert Syst. Appl.*, vol. 40, pp. 5501–5506, Oct. 2013.
[65] N. K. Treadgold and T. D. Gedeon, "The SARPROP algorithm: A simulated annealing enhancement to resilient back propagation," in *Proc. Int. Panel Conf. Soft Intell. Comput.*, Budapest, Hungary, 1996, pp. 293–298.
[66] B. E. Turvey, *Criminal Profiling: An Introduction to Behavioral Evidence Analysis*. Amsterdam, The Netherlands: Elsevier, 2011.
[67] *Criminal Intelligence: Manual for Analysts*, United Nations Publications, United Nations Office on Drugs and Crime, Vienna, Austria, 2011.
[68] D. T. Wagner, A. Rice, and A. Beresford, "Device analyzer: Understanding smartphone usage," in *Proc. 10th Int. Conf. Mobile Ubiquitous Syst., Comput., Netw. Services*, Tokyo, Japan, Dec. 2013, pp. 195–208.
[69] B. M. Wilamowski and H. Yu, "Improved computation for Levenberg–Marquardt training," *IEEE Trans. Neural Netw.*, vol. 21, no. 6, pp. 930–937, Jun. 2010.
[70] J.-H. Xia, R. Rusli, and A. S. Kumta, "Feedforward neural network trained by BFGS algorithm for modeling plasma etching of silicon carbide," *IEEE Trans. Plasma Sci.*, vol. 38, no. 2, pp. 142–148, Feb. 2010.
[71] H. C. Yuan, F. L. Xiong, and X. Y. Huai, "A method for estimating the number of hidden neurons in feed-forward neural networks based on information entropy," *Comput. Electron. Agricult.*, vol. 40, nos. 1–3, pp. 57–64, 2003.

KONSTANTIA BARMPATSALOU received the B.Sc. (Diploma) degree in information systems engineering and the M.Sc. degrees in information systems management and information security from the Department of Information and Communication Systems Engineering, University of the Aegean, Samos, Greece. She is currently pursuing the Ph.D. degree with the Department of Informatics Engineering, University of Coimbra. Her research interests include concepts such as digital forensics, information security, networks security, and intelligent systems.



TIAGO CRUZ received the Ph.D. degree in informatics engineering from the Department of Informatics Engineering, University of Coimbra, in 2012. He has been an Assistant Professor with the Department of Informatics Engineering, University of Coimbra, since 2013. His research interests cover areas such as management systems for communications infrastructures and services, critical infrastructure security, broadband access network device and service management, Internet of Things, software-defined networking, and network function virtualization (among others). He is a Senior Member of the IEEE Communications Society.

EDMUNDO MONTEIRO is currently a Full Professor with the University of Coimbra, Portugal. He has more than 30 years of research experience in the field of computer communications, wireless networks, quality of service and experience, network and service management, and computer and network security. He participated in many Portuguese, European, and international research projects and initiatives. His publication list includes over 200 publications in journals, books, and international refereed conferences. He has co-authored nine international patents. He is a member of the Editorial Board of the *Springer Wireless Networks Journal* and is involved in the organization of many national and international conferences and workshops. He is a Senior Member of the IEEE Communications Society and the ACM Special Interest Group on Communications. He is also a Portuguese Representative in IFIP TC6 (Communication Systems).

PAULO SIMOES is currently a Professor with the University of Coimbra. His main research interests are security, network management, and critical infrastructure protection. He was a co-founder of two technological spin-off companies in these areas, and he has authored or co-authored over 150 journal and conference publications in these areas. He is involved in several European research projects. He is involved in several European-funded and industry-funded research projects, with both technical and management activities.