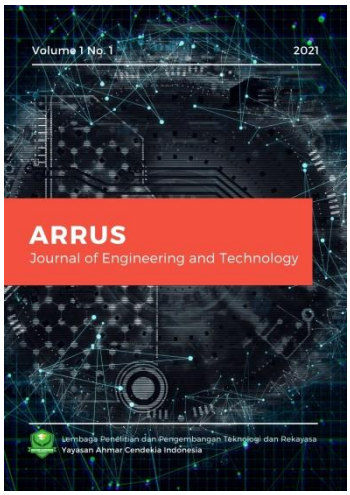
https://doi.org/10.35877/jetech613

OPEN ACCESS

ISSN : 2776-7914 (Print) / 2807-3045 (Online)



RESEARCH ARTICLE

Quest for Convergence Solution via Hybrid Genetic Algorithm Trained Neural Network Model for Metamorphic Malware Detection

Arnold Adimabua Ojugo[1*], Chris Obaro Obruche [2], Andrew Okonji Eboka[3]

[1]Department of Computer Science, College of Science, Federal University of Petroleum Resources Effurun, Delta State, Nigeria
[2]Research Assistant, Department of Computer Science, College of Science, Federal University of Petroleum Resources Effurun, Delta State, Nigeria
[3]Department of Network Computing, Coventry University, Priory Street Coventry CV1 5FB, United Kingdom

*Corresponding author: Arnold Adimabua Ojugo, Department of Computer Science, Federal University of Petroleum Resources Effurun, Delta State, Nigeria

E-mail: ojugo.arnold@fupre.edu.ng

Abstract: An unstable economy is rife with fraud. Perpetrated on customers, it ranges from employees' internal abuse to large fraud via high-value contracts cum control breaches that impose serious consequences to biz. Loyal employees may not perpetrate fraud if not for societal pressures and economic recession with its rationalization that they have bills to pay and children to feed. Thus, the need for financial institutions to embark on effective measures via schemes that will aids both fraud prevention and detection. Study proposes genetic algorithm trained neural net model to accurately classify credit card transactions. Compared, model used a rule-based system to provide it with start-up solution and it has a fraud catching rate of 91% with a consequent, false alarm rate of 9%. Its convergence time is found to depend on how close the initial solution space is to the fitness function, and for recombination and mutation rates applied.

Keywords: Malware, memetic algorithm, payload, deep learning, intrusion detection, convergence, metamorphic

1. INTRODUCTION

Malware has 3-modules: infect, trigger and payload. The Infect routine details the mechanism on how the malware modifies its host and copies itself unto the host machine. The trigger routine details how and when the malware delivers payload; while, the payload routine details what damage is achieved by the malware. Trigger and payload routines are optional as seen in fig 1. The infect pseudo-code. Subroutine *Infect* selects a target from M-targets to infect when run. *Select_target* details target selection criteria as same target should not be repeatedly selected; else, reveals presence of a virus. And, *Infect_code* performs actual infection by inserting its code into the target (Zink, 2009; Zakorzheysky, 2011; Ojugo and Yoro, 2021).

```
Def Virus():
        Infect()
        If Trigger() is TRUE then
        Payload is delivered()
```

```
Def Infect():
        Repeat M times()
          Target = Select_target()
                If no target() THEN
Return Infect code(target)
```

Figure 1. Sections 1 and 2 of Virus Pseudo-code

Malware self-replicates its codes onto a machine without the user's consent, and spreads by attaching a copy of itself to some part of program file. It attacks system resources and delivers

https://doi.org/10.35877/jetech613

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

a payload that aims to corrupt program, delete files, reformat disks, crash network, destroy critical data or embark on other damage to the host machine. Malwares are classified into (Ojugo and Yoro, 2020; Ye et al, 2008; Szor, 2005; Mishra, 2003; Orr, 2006; 2007):

- ✓ Simple virus that replicates itself when launched to gain control of a host machine by attaching copies of itself to another program as it spreads. It then transfers control back to host program. It can be detected through scan for a defined sequence of bytes, known as a signature
- ✓ Encrypted viruses scrambles its signature, making it unrecognizable at execution. Its decryption routine transfers control to its decrypted virus body so that each time it infects a new program, it makes copy of both the decrypted body and its related decryption routine. It then encrypts a copy and attaches both to a target system. It uses an encryption key to encrypt its body. As the key changes, it scrambles its body so that virus appears different from one infection to another. Such virus is difficult to detect via signature. Thus, antivirus must scan for a constant decryption routine.
- ✓ Polymorphic consists of a scrambled body, mutation engine and decryption routine. The decryption routine gains control to decrypt both its body and mutation engine. It then transfers control to the scrambled body to locate a new file to infect. It copies its body and mutation engine into RAM, and invokes its mutation engine to randomly generate new decryption routine to decrypt its body with little or no semblance to the previous routine. It then appends this newly encrypted body, a mutation engine and decryption routine to the newly infected file. Thus, the encrypted body and the decryption routine, varies from one infection to another. With no fixed signature and decryption routine, no two infections is alike.
- ✓ Metamorphic avoid detection by rewriting completely, its code each time it infects a new file. Its engine accomplishes this code obfuscation and metamorphism, which in most cases – is 90% of its assembly language codes (Singhal and Raul, 2012; Rabek et al, 2003).

1.1. Metamorphic: An Overview

Metamorphics transform its codes as they propagate to avoid detection by using obfuscation methods to alters its behaviour when it detects its execution within virtual machine (sandbox) as means to challenge a deeper analysis (Bolton and Head, 2002; Brause et al, 1999). Virus writer use the weaknesses in antiviruses, as limited to static and dynamic analysis, and attacks the following: (a) data flow, (b) control flow graph generations, (c) procedure abstract, (d) property verification, and (e) disassembly – all means to counter scans, to identify such metamorphic viruses (Burge and Shawe-Taylor, 2001). To mutate its code generation, metamorphic analyse their own codes and must re-evaluate the mutated codes generated (since complexity of transformation in the previous generation has a direct impact on its current state, how a virus analyses and transforms code in its current generation). Thus, they employ code conversion algorithm that helps them detect their own obfuscation and reordering (Delamaire and Abdou, 2009; Filiol, 2005). Thus, rather than use encryption, metamorphic can change their code structure and appearances while keeping its functionality intact. It does this via code obfuscation methods as in fig 2.

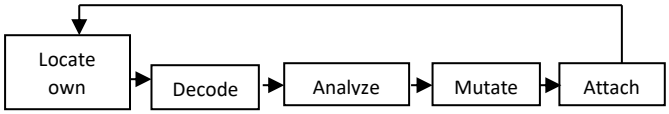


Figure 2. Distinct Signature of Metamorphic

Its engine reads in a virus executable, locates code to be transformed using its locate_own_code module. Each engine has its transformation rule that defines how a particular opcode or a sequence of opcodes is to be transformed. The decode module extracts the rules by disassembling the codes and passing it onto the next module. The ***analyze*** module analyses the current copy of the virus and determines what transformations must be

https://doi.org/10.35877/jetech613

OPEN ACCESS

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

applied to generate the next morphed copy. The mutate module performs the actual transformations by replacing an instruction (set) with the other its equivalent code; While, Attach module attaches the mutated or transformed copy to a host (Dheepa and Dhanapal, 2009; Duman and Ozcelik, 2011; Grimes, 2001; Hashemi et al, 2008; Sung et al, 2004; Desai, 2008; Walestein et al, 2007; Wong, 2006).

Khin (2019) A metamorphic engine may consist of: (a) internal disassemble to disassemble binary codes, (b) a shrinker replaces two or more codes with its single equivalent, (c) an expander replaces an instruction with many codes that performs same action, (d) a swapper reorders codes by swapping two/more unrelated codes, (e) a relocator assigns and relocate relative references such as jumps and call, (f) garbager (constructor) inserts whitespaces (do-nothing codes) to the program, and (g) cleaner (destructor) undoes actions of a garbager by removing whitespaces instructions. Some major feats of an effective metamorphic engine includes: (i) must be able to handle any assembly language opcode, (ii) shrinker and swapper must be able to process more than one instruction concurrently, (iii) garbager is used moderately, not to affect actual instructions, and (iv) swapper analyzes each instruction so as not to affect next instructions' execution (Kim et al, 2002; Maes et al, 2017; Malek et al, 2008; Venkatesan, 2006; Konstatinou, 2008).

1.2. Call for Papers and Word Count [Subheading 11pt, Garamond, Italic, Justified]

Metamorphic engine uses code obfuscation to yield morphed copies of original program. Obfuscated code is more difficult to understand and can generate different looking copies of a parent file as it operates on both control flow and data section of a program (Marane, 2011). Code obfuscation is achieved via (Nigrini, 2011; Borello and Me, 2008):

- Register Usage Exchange/Renaming – modifies the register data of an instruction without changing the codes itself, which remain constant across all morphed copies. Thus, only the operands changes.
- Dead Code inserts do-nothing (whitespace) codes that do not affect execution via a block or single instruction so as to change codes' appearance while retaining functionality.
- Subroutine Permutation aims to reorder subroutines so that a program of many subroutines can generate (n-1)! varied routine permutations, whose addition will not affect its functionality as this is not important for its execution.
- Equivalent Code Substitution replaces instruction with its equivalent instruction (or blocks). A general task can be achieved in different ways. Same feat is used in equivalent code substitution.
- Transposition/Permutation – modifies program execution order only if there is no dependency amongst instructions.
- Code Reorder inserts unconditional and conditional branch after each instruction (or block), and defines branching instructions to be permuted so as to change the programs' control-flow. Conditional branch is always preceded by a test instruction which always forces the execution of the branching instruction.
- Subroutine Inline/Outline is similar to dead code insertion in that subroutine call are replaced with its equivalent code as Inline inserts arbitrary dead code in a program; while outline converts block of code into subroutine and replace the block with a call to the subroutine. It essentially does not preserve any logical code grouping.

1.3. Study Motivation

Study is motivated thus (Stolfo et al, 2015; Ojugo and Eboka, 2021; Ojugo and Oyemade, 2021; Noreen et al, 2008):

1) With detection mechanisms broadly classified into: (a) signature-based scans for signature, and to evade it – virus makers create new virus strings that can alter their structure while keeping its functionality via code obfuscation method, and (b) code emulation creates sandbox or virtual machine, so that files are executed within it and

https://doi.org/10.35877/jetech613

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

scanned for virus. Once the virus is detected, it is no longer a threat – since it is running in controlled environment that limit damage to host machine. Thus, sharing malware detection implementation in details over public domain is quite unwise – as it will continually serve to further equip adversaries with adequate data required to evade detection.

2) The complex, dynamic and chaotic nature of fraudulent transactions using malware intrusion acts, and its range of complications as providing a backdoor to allow for other crime makes imperative and critical, early and accurate detection**.** Supervised detection alone via careful monitoring and management of network is insufficient as intruders often evade such as it often yields inconclusive results for ***unknown*** inputs. This leads to increased rate of false-positives and true-negatives. Our proposed model will effectively classify malware from genuine activities using the hybrid model as in Section II.
3) Unavailability of fraud datasets and its censored results – makes fraud detection techniques and studies difficult to assess. Dataset also consist of ambiguities, imprecision, noise and impartial truth that must be resolved via robust search in the bid to classify observations and expected values effectively as in Section(s) II and III respectively
4) Classification via predictive models is a complex and difficult task due to its chaotic and dynamic nature. Thus, we employ ***unsupervised*** model to resolve effectively and efficiently, statistical dependences and conflict imposed on the model by dataset used in approximating the data feats of interest.
5) Use of hill-climbing methods often has speed constraint imposed on it as the solutions are often trapped at local maxima. This is resolved with hybridization of statistical methods as in Section III/IV. Also, search for optimal via evolutionary heuristics can be quite cumbersome (though no one method yields better optimal than hybrids). Model must also resolve the statistical dependencies imposed on it by hybridization.
6) Search for optimal solution, may also yield overtraining and over-fitting of the model as it aims to find underlying probability of data feat(s) of interest. Also, improper selection of feats may also lead to over-parameterization.
7) Some model aim at a single suspicion score to globally classify statistical fraud. Studies show however, that some cases may be a result of true-negatives and false-positives scores as resolved in Section III.

Our proposed genetic algorithm trained neural net will employ unsupervised (improved) classification method that will help propagate observed data in model as it seeks data feats of interest to yield an output.. Evolutionary models have achieved great success in their enhancement for accurate prediction in its search for optimal solution, chosen from a set of possible solution space, to yield an output that is guaranteed of high quality and void of ambiguities. These models, further tuned can become robust and perform quantitative processing to ensure qualitative knowledge and experience, as its new language (Murad and Pinkas, 1999; Ojugo et al, 2013; Ojugo and Yoro, 2021; Kandeeban and Rajesh, 2007).

2. Literature Review

2.1. Data Gathering and Population

Our dataset contains 33,000 records of intrusion rules. Each record has 23-fields and our nondisclosure agreement prohibits us from revealing the details of the database schema as well as the contents of the data. But, it suffices to know that it is a common schema used by banks in Africa and Nigeria as part of the harmonization scheme. It contains information that banks deem important for identifying fraudulent transactions. The dataset was already classified into fraudulent or non-fraudulent classes. From records, 38.2% are fraud transactions (emanating from product transaction, asset misappropriation, corruption and financial statement fraud). The sampled data is for a 24-month period. Note that the number

https://doi.org/10.35877/jetech613

OPEN ACCESS

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

of fraud records for each month varies, and the fraud percentages for each month are different from the actual real-world distribution (Lakhotia et al, 2004).

2.2. Data Conditioning and Preprocessing

Here, we seek to know if (Perez and Marwala, 2011):

a. First, do we use the original data schema as is or do we condition (pre-process) the data by computing aggregate statistics or discretize certain fields? In our experiments, we removed several redundant fields from the original data schema. This helped to reduce the data size, thus speeding up the learning programs and making the learned patterns more concise. We have compared the results of learning on the conditioned data versus the original data, and saw no loss in accuracy.
b. Since the data has a skewed class distribution (20% fraud and 80% non-fraud), can we train on data that has (artificially) higher fraud rate and still compute accurate fraud patterns? And what is the optimal fraud rate in the training data? Pre-analysis experiments have shown that the training data with a 50% fraud distribution produces the best classifiers.
c. What percentage of the total available data do we use for our learning task? Most machine learning algorithms require the entire training data be loaded into the main memory. With our database very large, it is impractical. More importantly, we wanted to demonstrate that meta-learning can be used to scale up learning algorithms while maintaining the overall accuracy. In our experiments, only a portion of the original database was used for learning (details provided in the next section).
d. How do we validate and test our fraud patterns? In other words, what data samples do we use for validation and testing? In our experiment, the training data were sampled from earlier months, the validation data (for meta-learning) and the testing data were sampled from later months. The intuition behind this scheme is that we need to simulate the real world environment where models will be learned using data from the previous months, and used to classify data of the current month.
e. How do we evaluate a classifiers? Its accuracy is important but even a dummy algorithm can achieves 80% accuracy. For malware detection, its catching rate and false alarm rate are the critical metrics. A low catch-rate means that a large number of transactions will go through the system and cost users more money. Conversely, a high false alarm rate means that a large number of genuine rules will be blocked and human intervention is required to authorize such rules. Ideally, a cost function that takes into account both the True and False Positive rates should be used to compare the classifiers. For lack of cost information, we rank our classifiers using first the detection rate and the false alarm rate. Implicitly, we consider fraud catching rate as much more important than false alarm rate (Ojugo and Eboka, 2018a; 2018b, 2019; Tobiyama et al, 2016; Ojugo and Ekurume, 2021).

From original dataset, we prepared the data as suitable for use by the model by removing redundant fields. This helps to reduce the data size as well as speed up the learning heuristics, simplified the learning patterns and made the learned patterns more concise (as adapted from Stolfo et al, 2015). We also compared results of learning between our suitable data versus the original data, and saw no loss in accuracy. Also, observed data had a skewed distribution of 34% fraud and 66% non-fraud). We adopt 34% fraud class distribution as complete dataset (training data for fraud is always insufficient and we are not expecting an artificially, higher fraud rate to accurately compute suspicion score for fraud patterns). We also must determine ***suspicion*** score for each rule generated by the rule-based model in conjunction with the GA operators to help optimize functions for our training data. And though there are no rules for splitting data, we split it as 50% training, 25% cross-validation and 25% testing for fraud distribution, which also yielded the best classifier for the model. Thus, we demonstrate that even with outliers and noise in dataset and with imprecision and ambiguities applied at its

https://doi.org/10.35877/jetech613

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

input, model effectively classifies transactions into its proper classes. Thus, GANN effectively scales up learning algorithms void of over-parameterization, over-training and over-fitting of data feats; while maintaining overall performance accuracy (Ojugo and Eboka, 2019; 2020a; 2020b; 2020c; Xu et al, 2007).

2.3. Experimental Memetic Ensemble

The proposed model cum ensemble consists of these parts (Ojugo and Yoro, 2020; 2021; Wheeler and Aitkens, 2019):

a. Knowledgebase – consists of observed, historic structured data feats. The dataset is a record of fraudulent malware intrusion transactions stored and converted as fuzzy if-then ruleset using optimized membership functions. The rule-based system consists of *classifier* to propagate the IF-THEN rule values of selected data, enhanced them as predefined variables classification into intrusion types for fraud detection. Its houses the optimized universe discourse values as represented by fuzzy-if-then, linguistic variables (rule-based) as selected data feats.
b. Inference engine – consists of the memetic algorithm (i.e. the hybrid, rule-based genetic algorithm trained neural network model). The neural network is constructed using the Jordan network, and seeks to provide a self-learning ability, optimized by the CGA optimizer that recombines and mutates the rule-based fuzzy dataset to train and test the system so that it autonomously classify diabetes into its class types. Conversely, the Genetic Algorithm helps train the neural net so that combined – they effectively optimize our collated-answers within the tuned fuzzy rules values in other to yield a centralized, fuzzy-scaled function boundary in determining high/low degree membership function. Thus, the inference engine infers conclusion derived from genetic algorithm trained neural network from the selected data feats encoded as fuzzy-if-then conditions with possible outcomes and consequent action upon criteria being met.
c. Decision support– consists of the predicted output and the output database that is updated automatically in time as patients are diagnoses as long as it encounters and read sin new data. The decision support predicts system output based on the cognitive and the emotional filers as display by the output device. This is seen in fig 1.

The experimental ensemble is initialized with the if-then rules as individuals, whose fitness is computed. 30-individuals are then selected via tournament method as new pool. It then determines mating individuals to yield solutions. We use a multi-point crossover and mutation to help the network to learn all the dynamic and non-linear feats in the dataset (as feats of interest). With mutation, suspicion score for each rule between 1-to-30 is then randomly generated using Gaussian distribution corresponding to crossover points (all genes are from single parent). As new parents contribute the rest to yield new individuals whose genetic makeup is a combination of both parents, mutation is also applied to yield 3-random genes. These further undergo mutation and are then allocated new random values that still conform to the belief space. These random values will range between 0 and 1, which yields the suspicion score for each transaction as generated for each account holder (Syeda et al, 2002; Sylla and Wild, 2011; Vooshoghi et al, 2019).

The number of mutation applied depends on how far GA is progressed (how fit is the fittest individual in the pool), which equals fitness of the fittest individual divided by 2. New individuals replace old with low fitness so as to create a new pool. Process continues until individual with a fitness value of 0.8 is found – indicating that the solution has been reached (Ojugo and Ekurume, 2020; Ojugo and Otakore, 2018; 2020). Initialization and selection via ANN ensures that first 3-beliefs are met; mutation ensures fourth belief is met. Its influence function influences how many mutations take place, and the knowledge of solution (how close its solution is) has direct impact on how algorithm is processed. Algorithm stops when best

https://doi.org/10.35877/jetech613

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

individual has fitness of 0 (Ojugo and Otakore, 2021; Ojugo and Oyemade, 2021; Phua et al, 2007; Stolfo et al, 2000).

3. Result Findings and Discussion

3.1. Result Findings

For malware detection, the performance rating of any detection mechanism is in its fraud catching rate and its false alarm rate. These are critical metrics such that a low fraud catching rate implies that a large number of fraudulent transactions will go through the system; Thus, costing the banks a lot of money (and the cost will eventually be passed to the consumers. Also, a high false alarm rate implies that large number of legitimate transactions will be blocked by the detection system. Thus, supervised intervention, monitoring and management will then be required to authorize transactions. This will frustrate many customers, while also adding operational costs. Also, the malware detection rate is more important and critical than the false-alarm rate (true-negatives and false- positives). Ideally, a cost function that takes into account true-negatives and false-positive rates, should be used to compare the classifiers. For lack of cost data, we rank our classifiers using first the fraud catching rate as in table 1.

Table 1. Average Performance at each 1000th Generation

Generations	Average Fitness
0	0.0000
1000	0.6707
2000	0.7274
3000	0.6781
4000	0.6650
5000	0.8048

Performance is evaluated via computed values: mean square error, coefficient efficiency as well as on classification accuracy, false-positive and true-negative rates as in table 2 below (when compared to variant models):

Table 2. Model Convergence Performance Evaluation

Model	*MSE*	*COE*	*Classification Accuracy*	*False Positive*	*True Negative*
Fuzzy Rule			73%	27%	18.2%
ANN	0.230	0.310	0.82	10.96%	4.59%
Profile-HMM	0.134	0.280	0.90	12%	9.7%
Proposed Memetic Model	0.313	0.219	0.96	9%	5%

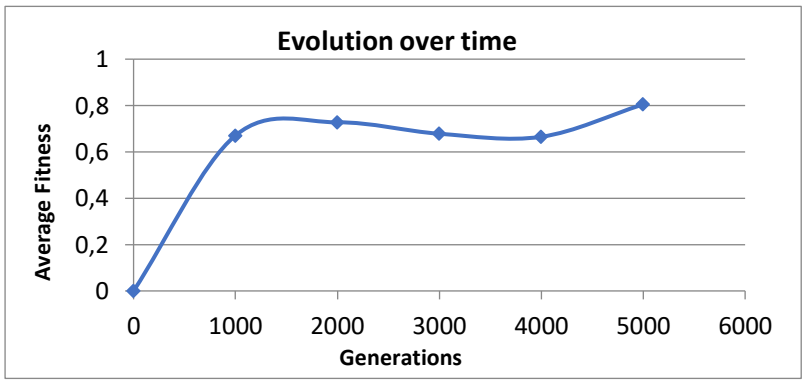


Figure 4. Evolution of Parameters in Time showing Convergence in Solution

After training and testing, compared to the models ANN, CGA and rule based system, the results are as follows: (a) ANN was run 24times and it took 223seconds to find solution after 98- iterations (best) and its fraud catching rate ranks at 76%. But, its demerit is that its solution is often trapped at local maxima, (b) GA was run 15-times to eradicate biasness and solution was found each time. It took 98seconds to find the solution after 123-iterations (best) and its

https://doi.org/10.35877/jetech613

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

fraud catching rate is 76%. Its convergence time depends on how close the initial population is to the solution as well as on the mutation applied to the individuals in the pool. Its demerit is that it seeks global optima (in this case, a single rule that can be applied to all transactions. This is in agreement with Ojugo and Yoro (2020; 2021). This would be appropriate if the transaction platforms are not considered as user are allowed to make transactions from various places – using varying devices that grants them access to their account at any point in time, and (c) CGANN with rule-based pre-processor hybrid was run 152times and its time varied between 29- and 245seconds to find solution after 102-iterations (best) and its fraud catching rate ranks at 91%. Its solution was made even closer using the fuzzy variable dataset as a pre-processor (agrees with Ojugo et al, 2014; 2015a; 2015b; Okobah and Ojugo, 2018).

3.2. Result Findings

Our hybrid memetic algorithm employed the fuzzy universe discourse linguistics and fuzzy system as a preprocessor. In the design, building and implementing if such hybrid – we took cognizance that genetic algorithm will help speed up the ANN to avoid it being trapped at local maxima as well as in region of multi-modal local maxima. This will enable the model yield robust optima in the shortest amount of time. The fuzzy system will help better represent variables and data values in the model. Hybrids have proven to be intelligent modules to transform transaction with adaptive results that provides potential model for fraud detection. Its generated rule set has an accuracy of 92%, sensitivity of 91%, and failure analysis (true-negative and false-positive rate) of 14% respectively. However, the extracted rules are sound and agree with outcome of relevant fraud detection norms and studies. Antivirus often impairs system performance, and incorrect decision may lead to security breach as it runs at the kernel of the operating system. If an antivirus uses heuristics, its success depends on the right balance between positives and negatives. Today, malware may no longer be executables. Macros can present security risk and antivirus heavily relies on signature-detection. Metamorphic and polymorphic viruses, evades and makes signature detection, quite ineffective.

4. Summary and Conclusion

Hybrids are quite difficult to implement and explore – even though they always yield optimal and better solutions. However, care must be employed during parameter selection to avoid over-fitting, over-parameterization and over-training. Also, the correctly formatted (explored) historic dataset must be encoded through the underlying algorithm's structured learning for robustness and code reuse as well as allow for model's adaptability and flexibility. This will in turn help to address the inherent issues of statistical dependencies imposed on the model by the various models fused for hybridization. However, proper encoding schemes must be selected to help resolve the conflicts in the data feats of interest – as most systems may not adequately highlight the implications of such in a multi-agent and multi-modal populated model. This is because the agents as they traverse the network or system – often can create their own behavioural rules on the dataset used – so that in most cases, they display results of complex chaos, non-linearity and dynamism (as expected) of the underlying probabilities of data feats of interest. To help curb this, we employed Cultural-GA, which ensures via its belief functions that all conditions to yield better generation is met with the processes of crossover and mutation applied.

References

Bolton, R and Hand, D., (2001). *Unsupervised Profiling Methods for Fraud Detection*. Credit Scoring and Credit Control VII, 22, pp149-178

Bolton, R.J and Hand, D.J., (2002). *Statistical fraud detection: a review*, Statistical Science, 17(3), pp235-255, 2002

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

Borello, J and Me, L., "Code obfuscation techniques for Metamorphics, 2008, [online]: available at www.springerlink.com/content/233883w3r2652537

Brause, R., T. Langsdorf, M. Hepp, (1999). *Neural Data mining for credit card fraud detection*, Proc. IEEE Int. Conf. Tools with Artificial Intelligence, pp. 103-106, 1999.

Burge, P and Shawe-Taylor, J., (2001). *An Unsupervised Neural, Network Approach to Profiling the Behaviour of Mobile Phone, Users for Use in Fraud Detection*. J. of Parallel and Distributed Computing 61: 915–925

Delamaire, L and Abdou, H., (2009). *Credit card fraud and detection techniques: a review*, Banks and Bank Systems, 4(2), pp57-67

Daoud, E and Jebril, I., (2008). Computer Virus Strategies and Detection Methods, International Journal of Open Problems Computational Mathematics, 1(2), [web]: www.emis.de/journals/IJOPCM/files/IJOPCM1.2.8.pdf

Desai, P., "Towards an undetectable computer virus, Masters Thesis, Department of Computer Science, San Jose State University, 2008

Dheepa, V and Dhanapal, R., (2009). *Analysis of Credit Card Fraud Detection Methods*, International Journal of Recent Trends in Engineering, 2(3), pp126-135.

Duman, E.M and Ozcelik, H., (2011). *Detecting credit card fraud by genetic algorithm and scatter search*. Expert Systems with Applications, 38: pp13057–13063

Fawcett, T., (1997). *AI Approaches to Fraud Detection and Risk Management*, AAAI Workshop. Technical Report WS-97-07, AAAI Press.

Filiol, E., "Computer Viruses: from Theory to Applications", New York, Springer, 2005, ISBN 10: 2287-23939-1.

Ghosh, S and Reilly, D.L., (1994) *Credit Card Fraud Detection with a Neural-Network*, Proc. 27th Int. Conf. System Sciences: Information Systems: Decision Support and Knowledge-Based Systems, 3, pp.621-630

Grimes, R., "Malicious Mobile Code: Virus Protection for Windows", O'Reilly and Associates, Inc., Sebastopol, CA, USA, 2001.

Hashemi,S., Yang, Y., Zabihzadeh, D and Kangavari, M., "Detecting intrusion transactions in databases using data item dependencies and anomaly analysis", Expert Systems, 2008, Vol. 25, No. 5, p460, doi:10.1111/j.1468-0394.2008.00467.

Kandeeban, S. S. and Rajesh, R. S., (2007): GA for framing rules for intrusion detection, J. Comp. Sci and Security., 7(11), ISSN:1738-7906, PP.285-290.

Khin, E.M., (2019). *Employing artificial intelligence to minimize internet fraud*. Int. Journal Cyber Society & Education, 2(1), pp.61-72, [web]: academic-journals.org/ojs2/index.php/IJCSE/article/viewFile/753/17

Kim, M.J and Kim, T.S., (2002). *A Neural Classifier with Fraud Density Map for Effective Credit Card Fraud Detection*, Proc. International Conf. Intelligent Data Eng. and Automated Learning, pp. 378-383, 2002

Konstantinou, E., "Metamorphic virus: analysis and detection", Technical report (RHUL-MA-2008-02), Dept. of Mathematics, Royal Holloway, University of London, 2008.

Lakhotia, A., Kapoor, A and Kumar, E.U., "Are metamorphic computer viruses really invisible?", 2004, Part 1, Virus bulletin, p5-7.

Maes, S., K. Tuyls, B. Vanschoenwinkel, B. Manderick, (2017). *Credit Card Fraud Detection*, Vrije Universiteit Brussel – Department of Computer Sci., Pleinlaan 2, B-1050, Belgium. [web]:personeel.unimaas.nl/k-tuyls/publications/papers/maenf02.pdf

Malek, W.M., K. Mayes, K. Markantonakis, (2008). *Fraud Detection and Prevention in Smart Card Based Environments Using Artificial Intelligence*. Int. Conf. CARDIS 2008, London, UK, September 8-11, 2008.

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

Marane, A., (2011). *Utilizing Visual Analysis for Fraud Detection, Understanding Link Analysis*, [web]: linkanalysisnow.com/2011/09/leveraging-visual-analytics-for.html

Minahan, T., (2013). *Fraud detection and prevention*. [web]: nebhe.org/info/pdf/tdbank_breakfast/Fraud_Prevention_and_Detection.pdf

Mishra, P., (2003). *Taxonomy of software unique transformations*, Available on [web]: www.cs.sjsu.edu/faculty/stamp/students/FinalReport.doc

Murad, U and Pinkas, G. (1999). Unsupervised Profiling for Identifying Superimposed Fraud. Proc. of PKDD99.

Nigrini, M. (2011). Forensic Analytics: Methods and Techniques for Forensic Accounting Investigation. Hoboken, NJ: John Wiley & Sons Inc. ISBN 978-0-470-89046-2. Available from [online]: http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470890460.html

Noreen, S., Ashraf, J and Svrenahak, K., "Malware detection using evolutionary models", International Journal of Virology, 2008, Vol. 23, No. 2, p123-132

Ojugo, A.A and Eboka, A.O,, (2015). An intelligent hunting profile for evolvable metamorphic malware", IEEE African Journal of Computing and ICT, 2015, 8(1-2), p181.

Ojugo, A.A and A.O. Eboka., (2018a). *Comparative evaluation for high intelligent performance adaptive model for spam phishing detection*, Digital Technology, Vol. 3, No.1: pp. 9-15, doi: 10.1269/dt-3-1-1, 2018

Ojugo, A.A and A.O. Eboka., (2018b). *Modeling market basket associative rule mining approaches using deep neural net*, Digital Technology, 3(1), pp.1–8, doi: 10.12691/dt-3-1-1

Ojugo, A.A and A.O. Eboka., (2019). *Signature-based malware detection using approximate Boyer Moore string matching algorithm*, International Journal of Mathematical Sciences and Computing, 3(5): pp49-62, doi: 10.5815/ijmsc.2019.03.05

Ojugo, A.A and A.O. Eboka., (2020a). *Memetic algorithm for short messaging service spam filter text normalization and semantic approach*, International Journal of Information & Communication Technology, 9(1), pp. 13 – 27, doi: 10.11591/ijict.v9i1.pp9-18

Ojugo, A.A and Eboka, A.O., (2020b). *Empirical evaluation on comparative study of machine learning techniques in detection of DDoS*, Journal of Applied Science, Engineering, Technology & Education, 2(1), pp18–27, doi: 10.35877/454RI.asci2192

Ojugo, A.A., Eboka, A.O., (2020c). *Modelling behavioral evolution as social predictor for the coronavirus contagion and immunization in Nigeria*, J. of Applied Sci., Engr., Tech. & Edu., 3(2): pp37–45, doi: 10.35877/454RI.asci130

Ojugo, A.A., Eboka, A.O., (2021). *Empirical Bayesian network to improve service delivery and performance dependability on a campus network*, International Journal of Artificial Intelligence, 10(3), pp623-635

Ojugo, A.A., Ekurume, E., (2021). *Predictive intelligence decision support model in forecasting of the diabetes pandemic using a reinforcement deep learning approach*, International Journal of Education and Mgt. Engineering, 2021, 11(2), pp.40-48, doi: 10.5815/ijeme.2021.02.05

Ojugo, A.A and Otakore, D.O., (2018). *Improved early detection of gestational diabetes via intelligent classification models: a case of Niger Delta*, J. of Computer Science & Application, 6(2), pp. 82-90, doi: 10.12691/jcsa-6-2-5

Ojugo, A.A., Otakore, O.D., (2020). *Forging an optimized Bayesian network model with selected parameter for detection of the Coronavirus in Delta State of Nigeria*, J. of Applied Sci., Engr., Tech. & Edu., 3(1): pp37–45, 2020, doi: 10.35877/454RI.asci2163

Ojugo, A.A and Otakore, D.O., (2021). *Forging optimized Bayesian network model with selected parameter for detection of Coronavirus in Delta State Nigeria*, Journal of Applied Science, Engineering, Technology & Education, 3(1): pp37–45, doi: 10.35877/454RI.asci2163

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

Ojugo, A.A., Oyemade, D.A., (2021) *Boyer Moore string-match framework for a hybrid short messaging service spam filtering technique*, IAES International Journal Artificial Intelligence, 10(3): pp519-527

Ojugo, A.A and Yoro, R.E., (2020). *Empirical solution for an optimized machine learning framework for anomaly-based network intrusion detection*, Technology Report of Kansai University, TRKU-13-08-2020-10996, 62(10): pp6353-6364

Ojugo, A.A and Yoro, R.E., (2021). *Forging a deep learning neural network intrusion detection framework to curb distributed denial of service attack*, International Journal of Electronics and Computer Engineering, Vol. 11, No. 2, pp 128-138

Ojugo, A.A., Allenotor, D. D.A. Oyemade., O. Longe., C.N. Anujeonye., (2015a). *Comparative stochastic study for credit-card fraud detection models*, African Journal of Computing and ICT, 8(1-2): pp15 –24, 2015.

Ojugo, A.A., Eboka, A., R.E. Yoro., M.O. Yerokun., F.N. Efozia., (2015b). *Framework design for statistical fraud detection*, Mathematics and Computers in Sciences and Engineering, 50: 176-182, ISBN: 976-1-61804-327-6.

Ojugo, A.A., Ben-Iwhiwhu, E., O.D. Kekeje., M. Yerokun., I. Iyawah., (2014). *Malware propagation on time varying networks: comparative study*, International Journal of Modern Education and Computer Science, 6(8), pp. 25-33, doi: 10.5815/ijmecs.2014.08.04

Ojugo, A.A., Emudianughe, J., Yoro, R.E., Okonta, E.O and Eboka, A.O., "Hybrid artificial neural network gravitational search algorithm for rainfall runoff", Progress in Intelligence Computing and Applications, 2013b, Vol. 2, No. 1, doi: 10.4156/pica.vol2.issue1.2, p22.

Okobah, I.P., Ojugo, A.A., (2018). *Evolutionary memetic models for malware intrusion detection: a comparative quest for computational solution and convergence*, IJCAOnline International Journal of Computing Application. 179(39), pp34–43

Orr, "The viral Darwinism of W32.Evol: an in-depth analysis of a metamorphic engine", 2006, [online]: available at http://www.antilife.org/files/Evol.pdf

Orr, "The molecular virology of Lexotan32: Metamorphism illustrated", 2007, [online]: www.antilife.org/files/Lexo32.pdf

Phua, C., D. Alahakoon, V. Lee, (2004). *Minority Report in fraud detection: classification of skewed data*, ACM SIGKDD Explorations Newsletter, 6(1), pp. 50-59, 2004

Phua, C., V. Lee, K. Smith, R. Gayler, (2007). *A comprehensive survey of data mining-based fraud detection research*, [web]: www.bsys.monash.edu.au/people/cphua/ .

Perez, M and Marwala, T., (2011). Stochastic optimization approaches for solving Sudoku, IEEE Transaction on Evol. Comp., pp.256–279.

Rabek, J., Khazan, R., Lewandowski, S., Cunningham, R., "Detection of injected, dynamic generated and obfuscated malicious code", In Proceedings of ACM Workshop on Rapid Malcode, 2003, p76.

Singhal, P and Raul, N., "Malware detection module using machine learning algorithm to assist centralized security in enterprise network", International Journal of Network Security and Applications, 2012, 4(1), doi: 10.5121/ijnsa.2012.4106, p61

Stolfo, S. J., Fan, D. W., Lee, W., Prodromidis, A and Chan, P. K. (2000). *Cost-Based Modeling for Fraud and intrusion detection: results from the JAM Project*, In Proc. DARPA Information Survivability Conf. and Exposition, vol. 2, pp. 130-144.

Sung, A., Xu, J., Chavez, P., Mukkamala, S., "Static analyzer of vicious executables", Proceedings of 20th Annual Computer Security Applications Conference, IEEE Computer Society, 2004, p326-334.

Syeda, M., Zhang, Y. Q. and Pan, Y. (2002). *Parallel Granular Networks for Fast Credit Card Fraud Detection*, Proc. IEEE Int'l Conf. Fuzzy Systems, pp. 572-577.

ISSN : 2776-7914 (Print) / 2807-3045 (Online)

Sylla BS, Wild CP. A million Africans a year dying from cancer by 2030: What can cancer research and control offer to the continent? Int J Cancer 2011; 130 (2): 245–250.

Szor, P., "The Art of Computer Virus Research and Defense", Addison Wesley Symantec Press. 2005, ISBN-10: 0321304543, New Jersey.

Tobiyama, S., Y. Yamaguchi., et al., (2016). *Malware detection with deep neural network using process behaviour*, *IEEE 40th Annual Computer Software and Applications Conf.*, Vol. 2, pp. 577-582, 2016

Vatsa, V., Sural, S. and Majumdar, A. K. (2005). *A game-theoretic approach to credit card fraud detection*, In. Proc. of Int. Conf. Information Systems Security, pp. 263-276.

Venkatesan, A., "Code obfuscation and metamorphic Virus Detection", Master thesis, San Jose State University, 2006, www.cs.sjsu.edu/faculty/students/ashwini_venkatesan_cs/report.doc

Voosoghi, R.B., Ghaffari, M and Razin, R., (2019). *Evaluation of the Efficiency of Adaptive Neuro Fuzzy Inference System in modeling of the Ionosphere Total Electron Content Time Series Case Study: Tehran Permanent GPS Station*, Journal of Geomatics Science and Tech., Vol. 8, no.4, Pp. 109-119, 2019

Walenstein, R., Mathur, M., Chouchane R., and Lakhotia, A., "The design space of metamorphic malware", In Proceedings of 2nd Int. Conference on Information Warfare, 2007, p243.

Wheeler, R., Aitken, S. (2019). *Multiple Algorithms for Fraud Detection Artificial intelligence Applications*, The University of Edinburg, Scotland, pp. 1-12, [web]: http://home.cc.gatech.edu/ccl/uploads/45/multiple-algorithms-for-fraud.pdf

Wong, W., "Analysis and Detection of Metamorphic Computer Viruses", Master's thesis, San Jose State University, 2006, http://www.cs.sjsu.edu/faculty/students/Report.pdf

Xu, J., Sung, A. H. & Liu, Q. (2007). *Behaviour Mining for Fraud Detection*, Journal of Research and Practice in Information Technology. 39(1), pp. 3–18

Ye, Y., Wang, D., Li, T and Ye, D., "Intelligent malware detection based on association mining", Journal of Computer Virology, 2008, Vol. 4, No. 4, p323–334, doi: 10.1007/s11416-008-0082-4.

Zakorzhevsky, E.R., "Monthly malware statistics", 2011,[online]:www.securelist.com/en/analysis/204792182/Monthly_Malware_Statistics_June_2011.

Zink, T., (2009). Network security algorithms, Konstanzer Online Publikationss-System, www.nbn-resolving.de/urn:nbn:de:bsz:352-175988