Evolutionary based Transfer Learning Approach to Improving Classification of Metamorphic Malware

Kehinde O. Babaagba[0000−0003−0786−2618] and Mayowa Ayodele[0000−0003−0854−4777]

School of Computing, Edinburgh Napier University, Edinburgh, UK
k.babaagba@napier.ac.uk,
Fujitsu Research of Europe, Slough, UK
mayowa.ayodele@fujitsu.com

Abstract. The proliferation of metamorphic malware has recently gained a lot of research interest. This is because of their ability to transform their program codes stochastically. Several detectors are unable to detect this malware family because of how quickly they obfuscate their code.It has also been shown that Machine learning (ML) models are not robust to these attacks due to the insufficient data to train these models resulting from the constant code mutation of metamorphic malware. Although recent studies have shown how to generate samples of metamorphic malware to serve as training data, this process can be computationally expensive. One way to improve the performance of these ML models is to transfer learning from other fields which have robust models such as what has been done with the transfer of learning from computer vision and image processing to improve malware detection. In this work, we introduce an evolutionary based transfer learning approach that uses evolved mutants of malware generated using a traditional Evolutionary Algorithm (EA) as well as models from Natural Language Processing (NLP) text classification to improve the classification of metamorphic malware. Our preliminary results demonstrate that using NLP models can improve the classification of metamorphic malware in some instances.

Keywords: Metamorphic Malware, Machine Learning, Evolutionary Algorithm, Transfer Learning, Natural Language Processing, Text Classification

1 Introduction

Detecting metamorphic malware have posed a serious challenge for antivirus engines among other detectors. This is because metamorphic malware comprise of a group of complex malware that transform their codes between generations. They do this using various obfuscation techniques to evade detection by malware detectors. Some of the mutation techniques they employ include but are not limited to junk code insertion (this inserts garbage code to the malware's program code which do not affect the behavior of the malware), instruction replacement (this substitutes valid instruction code with its equivalent without distorting the functionality of the program code), and instruction re-ordering (which distorts the flow of control of program by reordering its instruction code).

A few techniques have been employed in detecting this class of malware. The authors in [2] summarise the detection approaches to include Opcode-Based Analysis (OBA), Control Flow Analysis (CFA) and Information Flow Analysis (IFA). The various detection approaches depend on the kind of information employed when carrying out the analysis. Several other detection techniques have been suggested that involve the use of ML methods e.g Decision Trees (DT) in [10], Hidden Markov Models (HMM) in [37], Support Vector Machines (SVM) in [36] as well as a hybrid of both feature based and sequential ML models in [9].

However, ML techniques often involve using training data consisting of samples of known malware. Since metamorphic malware keep changing their code, there is usually insufficient training data, hence impeding ML model generality. Although previous work such as [6], [7] and [8] have provided methods for generating mutant samples of malware, this task involves several iterations to generate sufficient executable samples which can be very computationally expensive.

A good approach to tackle this problem would be to use transfer learning, a machine learning technique wherein the knowledge generated from a task is stored and reused in another task, often a related task [30]. Specifically, evolutionary based transfer learning where the training data are generated using an Evolutionary Algorithm (EA) [16]. EAs are population-based meta-heuristics that draws inspiration from processes occurring in biological evolution which guides a population to adapt towards a desired goal. Transfer learning is particularly useful in situations where there is lack of sufficient training data. This technique has been used in several domain such as NLP [34] and computer vision [20]. It has also been used in malware analysis, particularly, in image processing applications [33] among others. However, this has not been used in text classification within the context of malware analysis. Furthermore, in metamorphic malware detection, it has not been used to the best of our knowledge.

In [7] and [8], two EAs were used to generate training data which are used for the generation of the evolved malware mutants used in this work. These methods led to improved metamorphic malware detection within the context of feature based and sequential classifier [9]. The problem of limited training data was however noted. In this work, we use an evolutionary based transfer learning designed specifically for cases with limited data as a way of improving the classification and detection of metamorphic malware. Also, we seek to compare classification performance of NLP models.

In this work, we address the following research questions:

1. Can NLP language models be used in an evolutionary based transfer learning context to improve the classification of metamorphic malware?
2. Which of these NLP models provide the best classification performance for metamorphic malware?

We answer these questions by carrying out experiments that first compare the performance of ML models without the use of language models from NLP. Then, we test for performance improvement of the ML models using the NLP language models. We then analyse the NLP models to determine which one leads to the best improvement in classification scores of the ML models.

Summarily, the major contributions of this paper are as follows: Our experiments show that the use of NLP language models in an evolutionary based transfer learning context improves the classification of metamorphic malware in some instances. Also, we show that BERT language model has the best classification performance compared to the other language models tested.

The rest of the paper is structured as follows. Section 2 provides a review of related works. In Section 3, we describe the methodology of our research. Then we present and discuss our experimental settings in Section 4. Results are presented in Section 5. Section 6 summarises and concludes the paper, it also provides direction for future research.

2 Background

As earlier established, several approaches have been designed for metamorphic malware analysis and detection. These include statistical analysis of their binaries as in the work of [27] that used Linear Discriminant Analysis (LDA) and [39] that used Longest Common Subsequence (LCS). Other techniques include the use of control-flow graph matching [1], subroutine depermutation [19], code normalization [4] and similarity based approaches like structural entropy [11] as well as compression based classification [28] among others.

Machine learning techniques have also been used in classifying and detecting metamorphic malware. As earlier mentioned techniques such as HMM have been used by the authors in [5]. They analysed the performance of the HMM using 4 distinct compilers as well as handwritten assembly code with results showing the effectiveness of HMMs in the detection of metamorphic malware. The work in [10] employed decisions trees in metamorphic malware detection with classification results indicative of the reliability of decision trees in metamorphic malware classification. In [35], the authors used a single class SVM for detecting metamorphic malware with success. Furthermore, a combination of machine learning techniques have also been used for detecting metamorphic malware in [3]. Some authors [6], [7] and [8] have also generated metamorphic malware to serve as training data to improve the classification of machine learning models.

Transfer learning as a technique for use in instances of small training set or small labelled data has received a lot of research attention. There are application in a number of domains such as in medical application as seen in the work of [29] that modified the AlexNet [26] in order to detect Alzheimer's disease. It has also been used in bionformatics such as in [32] that used it for the study and prediction of associations in genotype-phenotype using Label Propagation Algorithm (LPA) [21]. In the transportation domain, it was applied in [15] to process similar images derived during varying conditions. It has also gained attention in NLP as seen in the work of [34].

Transfer learning has increasingly been used in malware detection. It has been used in computer vision for instance as in [13] where a computer vision based deep transfer learning was proposed for classifying static malware using knowledge from objects appearing in nature. Also, [33] used Deep Neural Network (DNN) built from the ResNet-50 architecture in classifying malicious software. The malware were converted to grayscale images. Then the DNN that had been trained previously on the ImageNet dataset is used in classifying the malicious samples. Similarly, the work of [12] built a deep learning model pre-trained on a large set of image data to improve the classification of malware. Transfer learning has also been employed in Generative Adversarial Network (GAN) settings as seen in the works of [23] and [24]. As in [24], their model comprised of a generator that created adversarial samples. The detector learns the characteristics of the malicious samples using a deep autoencoder (DAE). Prior to training the GAN, the DAE uses the learned features of malware to create data and transfers the learned information to improve the training of the GAN generator. Their method was shown to outperform other models designed for the same application. However, after exhaustive literature review, a study that employed evolutionary based transfer learning for metamorphic malware analysis and detection could not be found.

In this work, we use evolutionary based transfer learning of language models in NLP to improve the classification of metamorphic malware. We use evasive and diverse mutant variants of malware previously created in [7] and [8]. The effects of using these samples to improve current ML detection models including both feature based and sequential based models were analysed [9]. In this paper, we study if using these mutants in a model that employs transfer learning from NLP can improve their classification accuracy.

3 Methodology

In this section, we explain briefly the two EAs employed in the creation of the training set (details can be found in [7,8]). thereafter, we explain how we collect and process the data collected. We then explain the transfer learning models employed in our experimentation.

3.1 Creation of the Evolved Malware Mutants

Alg. 1 presents an EA originally proposed in [7]. The EA is a mutation only population based algorithm used in creating the evolved malware mutants. In Line 1, an initial population P consisting of n randomly generated mutants is created. These mutants are optimised for either the behavioral similarity between a variant and the original malware; the structural similarity between a variant and the original malware; the detection rate with respect to 63 detection-engines. Furthermore, we use the EA in [8] which employs MAP-Elites, a Quality Diversity

(QD) algorithm, to generate mutants that are structurally s and behaviorally b diverse to the original malware. Given each feature $< s, b >$, the algorithm seeks to find mutants associated to that feature that are as evasive as possible with results leading to the generation of more diverse mutants that retain their evasive ability. We ensure that the mutants created using both methods are still malicious by testing them against Droidbox[1]. This is a sandbox designed for the monitoring and dynamic analysis of mobile software. The sandbox works by executing the samples and then studying their behaviors by logging useful data relating to the sample such as its registry calls, process related operations among others.

To generate the final population of mutants, *max_iterations* generations of mutation steps are performed. During each generation, a new population R is created by randomly selecting k mutants from the initial population P (Line 4). In Line 6, one of three mutation types, which are, Garbage Code Insertion (GCI) (inserts a piece of junk code, e.g. a line number into the original program code), Instructional Reordering (IR) (adds a goto statement in the original program code that jumps to a label that does nothing) and Variable Renaming (VR) (renames a variable with another valid variable name in the original program code), is selected. In Line 7, a new mutant solution x_{new} is generated by performing mutation (using a randomly selected mutation type (*mut_type*) on the best solution in R (x_{best}). If the fitness of this new solution x_{new} is better than the worst solution in P, it replaces such solution, otherwise, x_{new} is discarded. At the end of *max_iterations* generations, the final population P is returned.

Algorithm 1 Evolutionary Algorithm [7]

```
1:  initialize population $P$ of size $n$.
2:  assign fitness $f(x)$ to each mutant $x \in P$
3:  while $max\_iterations$ not reached do
4:      $R \leftarrow$ randomly select $k$ variants from $P$
5:      $x_{best} \leftarrow \text{argmin}\,\{f(x), x \in R\}$
6:      $mut\_type \leftarrow$ select a mutation operator at random with uniform probability
7:      $x_{new} \leftarrow mutate(mut\_type, x_{best})$
8:      $fit_{new} \leftarrow f(x_{new})$
9:      $x_{worst} \leftarrow \text{argmax}\,\{f(x), x \in P\}$,
10:     $fit_{worst} \leftarrow f(x_{worst})$
11:     if $fit_{new} < fit_{worst}$ then
12:         replace $x_{worst}$ in $P$ with $x_{new}$
13:     end if
14: end while
15: return $P$
```

[1] Droidbox - `https://www.honeynet.org/taxonomy/term/191`

3.2 Data Collection and Processing

The samples used in this work comprise of Android malware which are archived as APK files. The main aim is to analyse metamorphic malware. However, due to the difficulty associated with the collection of these malware, we create mutant samples of existing popular malicious family as proxy which define prospective mutants. The samples comprise of both benign and malicious data.

The APK files comprise of 60 benign samples. These samples were collected from three categories namely; communication, entertainment and security. Equal number of samples were collected from each category resulting in 20 samples from each category. We chose these groups because they represent the behaviour of most Android clean files. The benign samples were collected from Google play store[1] (the samples were downloaded using Apkdownloader[2]) and Wondoujia play store[3].

The parent malware of the mutant variants of malware described in 3.1 were collected from Contagio Minidump[4] and Malgenome[5]. These comprise of three malware families and they are Dougalek[6], Droidkungfu[7] and GGtracker[8]. The three families were chosen based on their malicious payload and they belong to four groups described below:

1. Privilege Escalation: The complexity of the Android platform, owing to the fact that it comprises both Linux kernel and Android framework which have over 90 libraries, makes it prone to attacks in the form of privilege escalation. Droidkungfu [17], an Android malware family first discovered in May 2011, is an example of a malware family that uses privilege escalation. It is one of the families gotten from the MalGenome dump. It uses encryption to obfuscate its code in order to go undetected by detectors. It includes encrypted root exploits and malicious payloads that are in touch with C&C servers, from which they get instructions to be executed. This family of malware is considered in our analysis.
2. Remote control: This feature allows mobile malicious attackers gain remote control of the phone. Malware families that have this functionality are in communication with remote C&C servers. Droidkungfu is also an example of a malware family that uses remote control malicious payload and is considered in our analysis.

[1] Google Play - `https://play.google.com/store?hl=en`
[2] Apkdownloader -`https://apps.evozi.com/apk-downloader/`
[3] Wondoujia Play - `www.wandoujia.com`
[4] Contagio Minidump - `http://contagiominidump.blogspot.com/2015/01/android-hideicon-malware-samples.html`
[5] Malgenome - `http://www.malgenomeproject.org/`
[6] Dougalek - `https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/androidosdougalek.a`
[7] Droidkungfu - `https://www.f-secure.com/v-descs/trojan_android_droidkungfu_c.shtml`
[8] GGtracker - `https://www.f-secure.com/v-descs/trojan_android_ggtracker.shtml`

3. Financial charges: Some malicious attacks are launched to deliberately extort money from the users infected in form of financial charges. They subscribe users to premium services without proper authorisation, and in most cases the infected parties are unaware of such services. GGtracker [18] is an example of such a family of malware. It is one of the families in the MalGenome dump, and subscribes the infected users to various US premium services without their consent. It is also one of the families of malware studied.
4. Personal information stealing: There are also other malware families whose major goal is to collect information. This information could be on the infected user's account, contact list, text messages, among others. Malware families such as Dougalek [38] from the Contagio minidump and GGtracker, fall into this category and are analysed in our study.

The malicious samples also comprise of malware gotten from the web that belong to the aforementioned families collected from Contagio Minidump.

In order to get the features of the samples collected we carryout dynamic analysis of the samples using tools such as Strace[9] and MonkeyRunner[10]. Strace runs the samples in order to study its behavior and keeps track of each system call the samples make. It uses MonkeyRunner to execute the sample's main activity and MonkeyRunner is employed in simulating user interaction with the sample. This is then used to generate sequential features of the samples.

We use the log stored by Strace to derive each sample's sequential features. Thereafter, we generate a time-ordered system-calls list and this forms the feature vector of the samples.

3.3 NLP Language Models

In this section we describe the language models employed in this work. The language models selected comprise of some of the most recent and commonly used models in NLP. They are briefly explained below:

BERT [14] is an acronym for Bidirectional Encoder Representations from Transformers. It was created for the pretraining of deep representations that are bidirectional, from text that are not labelled by taking into consideration the contextual information of the text that is, by working out both the left and right context of the token. Consequently, the pre-trained BERT models can be easy adjusted and tuned with only an extra output layer to produce advanced models for a large number of NLP tasks. This model is pre-trained on a massive unlabelled text corpus which includes the whole of Wikipedia (this has about 2.5 billion words) and Book Corpus (this comprises of about 800 million words). After being tested on about 11 NLP tasks, it produces novel state-of-the-art results such as improving the GLUE score by 7.7%, the MultiNLI accuracy by 4.6%, the SQuAD v2.0 Test F1 by 5.1%, among others.

[9] Strace - `https://linux.die.net/man/1/strace`
[10] Monkeyrunner - `https://developer.android.com/studio/test/monkey`

As illustrated in Fig. 1, BERT model comprises both pre-training and fine-tuning steps. The pre-training task occurs with the training of the model on instances that are unlabelled for various distinct pre-training tasks. The fine-tuning process on the other hand, begins with the initialization of the model with pre-trained parameters. Then, using the labelled data derived from the downstream tasks, the parameters are fine-tuned. BERT uses $O(n^2)$ time and space with regard to the length of the sequence.

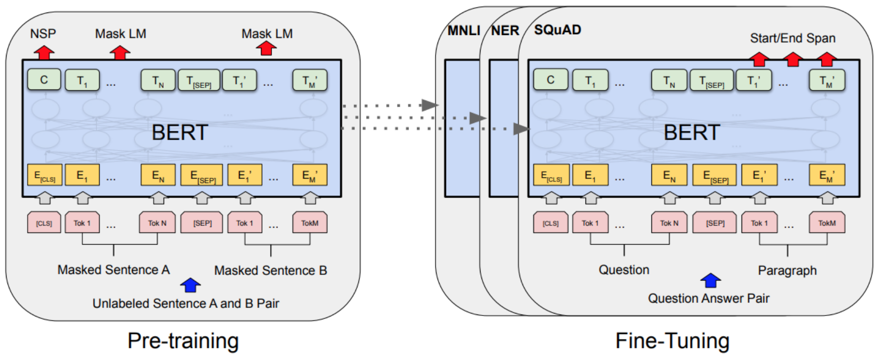

Fig. 1. A BERT model illustrating its pre-training and fine-tuning tasks [14]

GloVE [31] A number of models that use unsupervised techniques to understand word representations often rely on and use word occurrences statistics in a corpus to learn from word representations. However, a number of unanswered questions exist regarding how meaningful these statistics are as well as if the word vectors generated from them provide meaningful representations. The GloVe (Global Vectors) model was presented as an unsupervised learning algorithm used to represent words which directly captures the global corpus statistics in the model. It generates vector representations for words and trains on a composite of global word-word co-occurrence statistics from a corpus. It has been shown to produce representations with striking and meaningful linear substructures of the word vector space.

An example can be seen in making a quantitative distinction between man and woman as seen in Fig. 2. To do that, an association has to be built beyond one number to the pair of words by a model, for instance through the vector difference between their word vectors. In such an example, GloVe is well suited for computing such vector differences such that the meaning derived from the collocation of the two words is maximally represented.

FastText [22] This model was generated by researchers at Facebook AI Research (FAIR) lab to serve as a library for learning word representations as

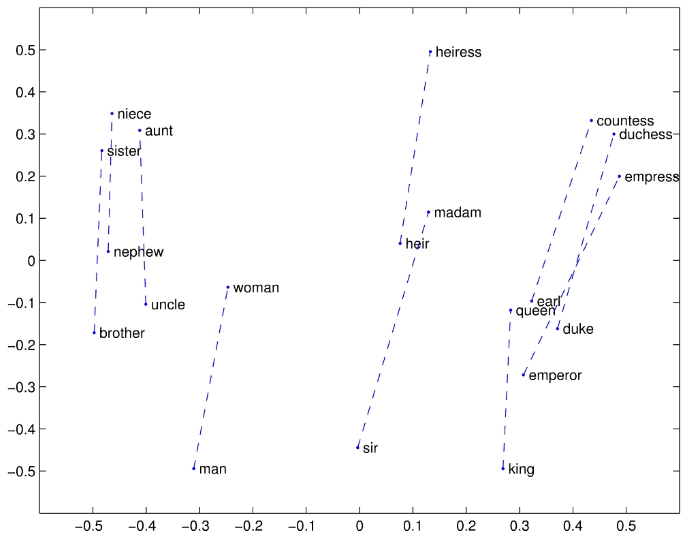

Fig. 2. Linear substructure for quantitatively distinguishing between man and woman using GloVe model [31]

well as sentence classification more effectively. Unlike other word vectors that consider each word as the lowest unit in which we are seeking to find its representation, FastText considers each word as a n-grams of character, in which n can take values from 1 to the word length. It is beneficial in that it can discover the vector representation for uncommon words as these words can be split into character n-grams. It incorporates pre-trained language models learned in over 157 different languages and including the whole of Wikipedia.

For complex and rare words that would have been difficult to represent, other than return a zero vector or a random vector with low magnitude, FastText will split those words into character n-grams and use the vectors of the generated character n-grams to produce the final word vector. This kind of embedding has been shown to outperform other embeddings particularly on smaller data-sets and its architecture is given in Fig. 3.

4 Experimental Settings

The parameters used by the EA is the same as one in [7] and [8] and presented in Table 1.

Our experiments were implemented using Scikit-learn libraries for Python, including the use of the Keras library[11]. The models (explained in Section 3.3 of the paper) and their hyper-parameters were empirically tuned. As a result of its documented success in terms of its accuracy and computational power, "Adam"

[11] Keras - `https://github.com/fchollet/keras`

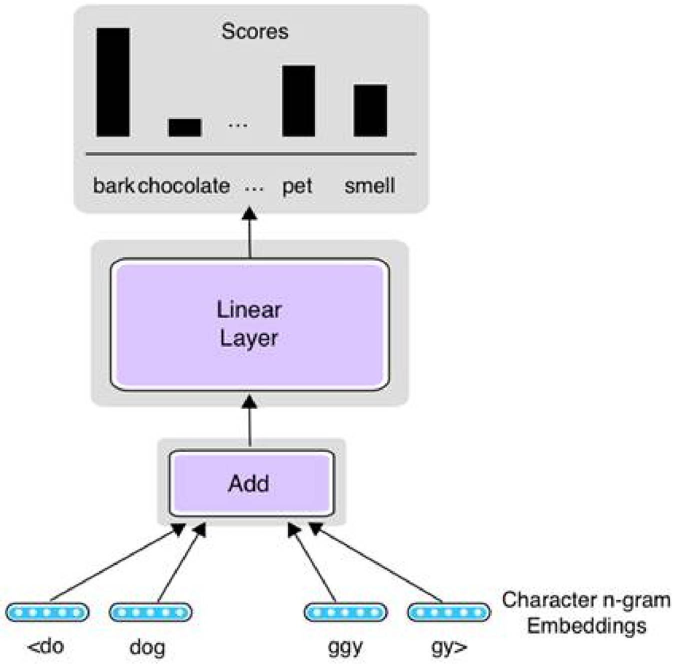

Fig. 3. FastText Architecture [22]

Table 1. Evolutionary based Parameter Settings

EA	Settings	
	EA	MAP-Elites
Bootstrap	NA	20
Selection	Tournament	Random
Population Size	20	NA
Iterations	120	120
Mutation Rate	1	1

optimiser [25] was employed. The binary cross entropy function was used as the loss function (this function was chosen as our classification is binary). Moreover, as our problem is a classification problem, we employ a Dense output layer comprising of one neuron with a sigmoid activation function. We employed a batch size of 6 so as to space out the updates of weight. The model was fitted using just four epochs as it speedily over-fitted the problem.

The training set comprises of 60 benign and 60 malicious samples. The 60 benign samples comprise of 20 entertainment applications, 20 security applications and 20 communication applications. The 60 malicious samples comprise of 20 malware from the Dougalek family, 20 malware from the Droidkungfy family and 20 malware from the GGtracker family. We will refer to the data combination as 6020combo from here on.

Also we consider increasing the malicious samples for training by considering 60 benign samples and 157 malicious samples (50 from Dougalek family, 55 from the Droidkunfu family and 52 from the GGTracker family). We will refer to this increased data combination as 6050combo from here on.

For testing, we use dataset comprising of 27 benign samples, 23 malicious samples (10 dougalek family, 5 droidkunfu family and 8 ggtracker family) for the 6050combo. For the 6020combo, we use a dataset consisting of 27 benign samples, 16 malicious samples (10 dougalek family, 3 droidkunfu family and 3 ggtracker family).

The approach proposed in this paper, provides a robust solution for detecting novel mutants of malware which represent the type of malware found in real environments.

5 Results

In this section, we analyse results based on the experimental settings described in the previous section. We particularly provide answers to our research questions in the subsections below.

5.1 Can NLP language models be used in a evolutionary based transfer learning context to improve the classification of metamorphic malware?

To answer our first research question, we conduct experiments with and without the use of the NLP language models and observe if there was an improvement in the classification accuracy and F1 score by reason of using the language models. This was done for the 6020combo and the 6050combo as shown in Tables 2 and 3.

We see that when we do not use a language model, we get an accuracy of 0.63 for the 6020combo and an accuracy of 0.54 for the 6050combo. Although the same results are obtained when we use the GloVe and FastText models, we see that for both the 6020combo and 6050combo data, we get an improved

Table 2. Comparing Accuracy obtained on the test sets for the 6020combo and 6050combo models using No language Model, BERT, FastText and GloVE language models

Test Sets	Accuracy			
	No Language Model	BERT	FastText	GloVE
6020combo	0.63	**0.93**	0.63	0.63
6050combo	0.54	**0.90**	0.54	0.54

classification score of 0.93 and 0.9 respectively using the BERT model. There is at least one model — BERT that results in an improved classification accuracy.

Similarly, when F1 Score - which computes the harmonic mean of precision and recall is employed as an evaluation metric, we notice a similar trend showing that in two instances i.e., BERT (0.91 for the 6020combo and 0.9 for the 6050combo) and GloVE (0.8 for the 6020combo and 0.7 for the 6050combo) the F1 Score is higher when an NLP language model is employed than when no language model is used which results in an F1 Score of 0.5 for the 6020combo and 0.4 for the 6050combo. It is important to note that the BERT model has been shown to be significantly better than other language models when smaller data-sets are involved [14].

Table 3. Comparing F1 Score obtained on the test sets for the 6020combo and 6050combo models using No language Model, BERT, FastText and GloVE language models

Test Sets	F1 Score			
	No Language Model	BERT	FastText	GloVE
6020combo	0.5	**0.91**	0.5	0.8
6050combo	0.4	**0.9**	0.4	0.7

5.2 Which of these NLP models provide the best classification performance for metamorphic malware?

In this section, we compare the performance of the three language models to see which one produces the best classification accuracy and F1 Score. From Table 2, we see that the classification accuracy of both FastText and GloVe models are the same for both the 6020combo and 6050combo models. However, we see that the BERT model performs significantly better than the other two models producing an accuracy of 0.93 and 0.9 for the 6020combo and 6050combo data respectively.

Table 3 also shows that compared to the other models BERT has a better F1 Score of 0.91 for the 6020combo and 0.9 for the 6050combo. It is interesting to note that when we use the F1 Score, GloVE (0.8 for the 6020combo and 0.7 for the 6050combo) outperforms FastText (0.5 for the 6020combo and 0.4 for the 6050combo) for both the 6020combo and 6050combo.

6 Conclusion

We have established that metamorphic malware represent a difficult class of malware to detect due to the way they change their codes stochastically. Another problem with detecting these malware class particularly using ML models is that there is insufficient training data for ML models to learn from. Generating these data is very time consuming and computationally expensive.

In this paper, we have presented an approach to address the aforementioned problem that employs an evolutionary based transfer learning method to improve the classification of metamorphic malware. The results show that the use of BERT model leads to better classification accuracy and F1 Score compared to when a language model is not used. Furthermore, we demonstrate that the use of BERT model also yields the best accuracy and F1 Score on both data tested as compared to the other two language models employed.

Future work could compare more NLP models as well as use transfer learning from other application areas for improved classification and detection of metamorphic malware.

References

1. Alam, S., Traore, I., Sogukpinar, I.: Annotated control flow graph for metamorphic malware detection. The Computer Journal **58**(10), 2608–2621 (Oct 2015). https://doi.org/10.1093/comjnl/bxu148
2. Alam, S., Traore, I., Sogukpinar, I.: Current trends and the future of metamorphic malware detection. In: Proceedings of the 7th International Conference on Security of Information and Networks. pp. 411–416. SIN '14, ACM, New York, NY, USA (2014)
3. Alazab, M., Venkatraman, S., Watters, P., Alazab, M.: Zero-day malware detection based on supervised learning algorithms of api call signatures. In: Proceedings of the Ninth Australasian Data Mining Conference - Volume 121. pp. 171–182. AusDM '11, Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2011), http://dl.acm.org/citation.cfm?id=2483628.2483648
4. Armoun, S.E., Hashemi, S.: A general paradigm for normalizing metamorphic malwares. In: 2012 10th International Conference on Frontiers of Information Technology. pp. 348–353 (Dec 2012). https://doi.org/10.1109/FIT.2012.69
5. Austin, T.H., Filiol, E., Josse, S., Stamp, M.: Exploring hidden markov models for virus analysis: A semantic approach. In: 2013 46th Hawaii International Conference on System Sciences. pp. 5039–5048 (Jan 2013). https://doi.org/10.1109/HICSS.2013.217

6. Aydogan, E., Sen, S.: Automatic generation of mobile malwares using genetic programming. In: Mora, A.M., Squillero, G. (eds.) Applications of Evolutionary Computation. pp. 745–756. Springer International Publishing, Cham (2015)
7. Babaagba, K.O., Tan, Z., Hart, E.: Nowhere metamorphic malware can hide - a biological evolution inspired detection scheme. In: Wang, G., Bhuiyan, M.Z.A., De Capitani di Vimercati, S., Ren, Y. (eds.) Dependability in Sensor, Cloud, and Big Data Systems and Applications. pp. 369–382. Springer Singapore, Singapore (2019)
8. Babaagba, K.O., Tan, Z., Hart, E.: Automatic Generation of Adversarial Metamorphic Malware Using MAP-Elites. In: P.A. Castillo et al (ed.) 23rd European Conference on the Applications of Evolutionary and bio-inspired Computation. pp. 1–16. Springer-Verlag New York, Inc., Seville (2020)
9. Babaagba, K.O., Tan, Z., Hart, E.: Improving classification of metamorphic malware by augmenting training data with a diverse set of evolved mutant samples. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–7. IEEE (2020)
10. Bashari Rad, B., Masrom, M., Ibrahim, S., Ibrahim, S.: Morphed Virus Family Classification Based on Opcodes Statistical Feature Using Decision Tree. In: Abd Manaf, A., Zeki, A., Zamani, M., Chuprat, S., El-Qawasmeh, E. (eds.) Informatics Engineering and Information Science. pp. 123–131. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
11. Baysa, D., Low, R.M., Stamp, M.: Structural entropy and metamorphic malware. Journal in Computer Virology **9**(4), 179–192 (2013). https://doi.org/10.1007/s11416-013-0185-4
12. Bhodia, N., Prajapati, P., Troia, F.D., Stamp, M.: Transfer learning for image-based malware classification. CoRR **abs/1903.11551** (2019), `http://arxiv.org/abs/1903.11551`
13. Chen, L.: Deep transfer learning for static malware classification. CoRR **abs/1812.07606** (2018), `http://arxiv.org/abs/1812.07606`
14. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), `http://arxiv.org/abs/1810.04805`
15. Di, S., Zhang, H., Li, C., Mei, X., Prokhorov, D., Ling, H.: Cross-domain traffic scene understanding: A dense correspondence-based transfer learning approach. IEEE Transactions on Intelligent Transportation Systems **19**(3), 745–757 (2018)
16. Eiben, A.E., Smith, J.E.: What is an Evolutionary Algorithm? In: Introduction to Evolutionary Computing, pp. 15–35. Springer Publishing Company, Incorporated (2003)
17. F-Secure: Trojan:Android/DroidKungFu.C (2019), `https://www.f-secure.com/v-descs/trojan_android_droidkungfu_c.shtml`
18. F-Secure: Trojan:Android/GGTracker.A (2019), `https://www.f-secure.com/v-descs/trojan_android_ggtracker.shtml`
19. Fiñones, R.G., Fernandez, R.: Solving the metamorphic puzzle. Virus Bulletin pp. 14–19 (2006), `https://www.virusbulletin.com/virusbulletin/2006/03/solving-metamorphic-puzzle/`
20. Gao, J., Ling, H., Hu, W., Xing, J.: Transfer learning based visual tracking with gaussian processes regression. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 188–203. Springer International Publishing, Cham (2014)
21. Hwang, T., Kuang, R.: A heterogeneous label propagation algorithm for disease gene discovery. In: Proceedings of the 2010 SIAM International Conference on Data Mining. pp. 583–594. SIAM (2010)

22. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. pp. 427–431. Association for Computational Linguistics, Valencia, Spain (Apr 2017), `https://aclanthology.org/E17-2068`
23. Kim, J.Y., Bu, S.J., Cho, S.B.: Malware detection using deep transferred generative adversarial networks. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.S.M. (eds.) Neural Information Processing. pp. 556–564. Springer International Publishing, Cham (2017)
24. Kim, J.Y., Bu, S.J., Cho, S.B.: Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. Information Sciences **460-461**, 83 – 102 (2018). https://doi.org/https://doi.org/10.1016/j.ins.2018.04.092
25. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)
27. Kuriakose, J., Vinod, P.: Ranked linear discriminant analysis features for metamorphic malware detection. In: 2014 IEEE International Advance Computing Conference (IACC). pp. 112–117 (Feb 2014). https://doi.org/10.1109/IAdCC.2014.6779304
28. Lee, J., Austin, T.H., Stamp, M.: Compression-based analysis of metamorphic malware. Int. J. Secur. Netw. **10**(2), 124–136 (Jul 2015). https://doi.org/10.1504/IJSN.2015.070426
29. Maqsood, M., Nazir, F., Khan, U., Aadil, F., Jamal, H., Mehmood, I., Song, O.Y.: Transfer Learning Assisted Classification and Detection of Alzheimer's Disease Stages Using 3D MRI Scans. Sensors (Basel, Switzerland) **19**(11), 2645 (jun 2019). https://doi.org/10.3390/s19112645
30. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010)
31. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
32. Petegrosso, R., Park, S., Hwang, T.H., Kuang, R.: Transfer learning across ontologies for phenome–genome association prediction. Bioinformatics **33**(4), 529–536 (nov 2016). https://doi.org/10.1093/bioinformatics/btw649
33. Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., de Geus, P.: Malicious software classification using transfer learning of resnet-50 deep neural network. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 1011–1014 (2017)
34. Ruder, S., Peters, M.E., Swayamdipta, S., Wolf, T.: Transfer learning in natural language processing. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials. pp. 15–18. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-5004
35. Sahs, J., Khan, L.: A machine learning approach to android malware detection. In: 2012 European Intelligence and Security Informatics Conference. pp. 141–147 (Aug 2012). https://doi.org/10.1109/EISIC.2012.34
36. Sahs, J., Khan, L.: A Machine Learning Approach to Android Malware Detection. In: 2012 European Intelligence and Security Informatics Conference (2012)

37. Toderici, A.H., Stamp, M.: Chi-squared Distance and Metamorphic Virus Detection. J. Comput. Virol. **9**(1), 1–14 (feb 2013)
38. TRENDMICRO: ANDROIDOS_DOUGALEK.A (2012), `https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/androidos_dougalek.a`
39. Vinod, P., Vijay, L., Singh, G.M., Phani, K.G., S., C.Y.: Static cfg analyzer for metamorphic malware code. In: Proceedings of the 2Nd International Conference on Security of Information and Networks. pp. 225–228. SIN '09, ACM, New York, NY, USA (2009). https://doi.org/10.1145/1626195.1626251