Journal of Information Systems Engineering and Management
2025, 10(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Deep Learning Approach for Android Malware Detection Using Mixed Bytecode Images and Attention-Based ResNet

S. Girirajan[1], Cindhe Ramesh[2], Mala Saraswat[3], Vimmi Kochher[4], Kodge B. G[5], Dr Brajesh Kumar Singh[6], Dr Saroo Raj R B[7]

[1]Assistant Professor, Department of Computing Technologies, SRM institute of science and technology, Kattankulathur, Tamil Nadu, India, girirajans.cse@gmail.com

[2]Professor, Department of Computer Science and Engineering, CVR College of Engineering, Vastunagar, Mangalpally(v), Ibrahimpatnam (m),R.R.Dist., Telangana-501510, hmcr.ramesh@gmail.com

[3]Associate Professor, SCSET, Bennett University, Greater Noida, U.P, malasaraswat@gmail.com

[4]Research scholar, Department of Computer science and engineering, Starex University, Gurugram, v.kochher1990@yahoo.com

[5]Associate professor, Dept of Computer Science, School of Science, GITAM University, Hyderabad, TS, India. kodgebg@gmail.com

[6]Associate Professor, Electronics and Communication Engineering, Galgotia College of Engineering and Technology, brajeshsingh.dce@gmail.com

[7]Professor, Department of Computer Science and Engineering, Saveetha School of Engineering , Saveetha Institute of Medical and Technical sciences, Saveetha University,Chennai, Tamilnadu,India, saroorajrb.sse@saveetha.com

Corresponding author mail: [2]hmcr.ramesh@gmail.com, [3]malasaraswat@gmail.com, [5]kodgebg@gmail.com

ARTICLE INFO

Received: 15 Oct 2024

Revised: 10 Dec 2024

Accepted: 22 Dec 2024

ABSTRACT

In an era, where apps are on the rise and taking over almost everything we do on our devices, detecting Android malware becomes of utmost importance. This work proposes a deep learning framework that can overcome those constraints in signature-based detection while against new and ever evolving flavors of malware. This technique consists in converting an Android application bytecode, into set of images which represent dynamic (behaviour and structure) features the program. It is these images that will be are basis for applying complex image classification methods effectively to detect malware. At the heart of this framework is an attention-amplified ResNet, one of the revolutionary convolutional neural network structures that can be used for image recognition. The introduction of the attention mechanism permits the model to pay more (less) attention on certain portions in the corresponding bytecode images, which helps boost its capacity for distinguishing benign and malicious applications with higher focus weights. The approach increases the generalization properties of a model to varying malware families and improves its adversarial robustness for withstanding new evasion strategies by using bytecode images that are mixed, i.e., original as well as augmented versions. Experiments on a large dataset of Android applications show that inaccuracy, precision and recall performed jointly polished than current models dosing malware detection. It turns out that the attention mechanism plays an important role for improving detection performance, especially on heavily obfuscated malware. Furthermore, the application of mixed bytecode images reduces false positives and increases accuracy to a level where such models can be deployed in practice. Our work provides support for the use of image-based analysis and deep learning in Android malware detection. This attention-based ResNet model will allow us to develop new, more complex security solutions that are better suited for today's ever-changing mobile threat environment.

Keywords: android, malware, bytecode, images, amplified, recognition, classification, generalization, model, method, dynamic

INTRODUCTION

With Android apps blossoming like never before, the trend shows no sign of slowing down and users have their hands on many other android applications that change their operating environment providing them with a completely new

level of convenience. Android malware has grown into a critical problem for users and enterprises; although, this technological progress poses considerable security challenges [1]. Increasing frequency, and the increasing sophistication of malware attacks have demonstrated a deficiency in current methods based on signatures for detection Piquet The rapid evolution of malicious behaviour has illustrated that relying upon patterns is not enough.

The main approach for malware detection is based on traditional methods that depend mainly signature-based, the algorithm starts by configuring databases of known signatures and then files are compared against these characteristics [2][3]. These are great for discovering known threats, but do little to detect new or edited versions of malware. Moreover, the fact that malware encounters too many transformations such as obfuscation and polymorphism which causes difficulty in signature-based detection mechanisms acts against this approach [4][5]. While these approaches do manage to defend against old basic hazards, they often fail when it comes to offering all-around safety in opposition up-to-the-minute threats—increasing the exposure of their networked systems towards future risks.

One alternative approach is heuristic-based methods, which uses predetermined rules and behavioural patterns to detect potential threats [6]. Though these methods are much more flexible compared to signature-based techniques, they have their own major challenges. The heuristic methods yield very high amounts of false positives, as they tend to match benign applications with the malicious ones as per their heuristics rules [9]. In addition, they may be ineffective in detecting bio-malware (i.e., a type of malware written in any kind of script language) because the payload part could use an encrypted code and can also deploy polymorphic evasion techniques [8]. This should definitely seem that the way signature-based nor heuristic-based methods are blocked has inherent limitations.

Machine learning and deep-learning approaches have recently been proposed as promising solutions to improve the malware-detection capabilities [9][10]. Such techniques can potentially go beyond the constraints of classical methods by using learned patterns on extensive data sets and adjusting to new, emerging threats. In particular, deep learning has lately enjoyed tremendous success in numerous image classification challenges by its ability to learn hierarchical features directly from raw data (11). This success motivates the research community to explore deep learning for malware detection, towards better accuracy and generalization.

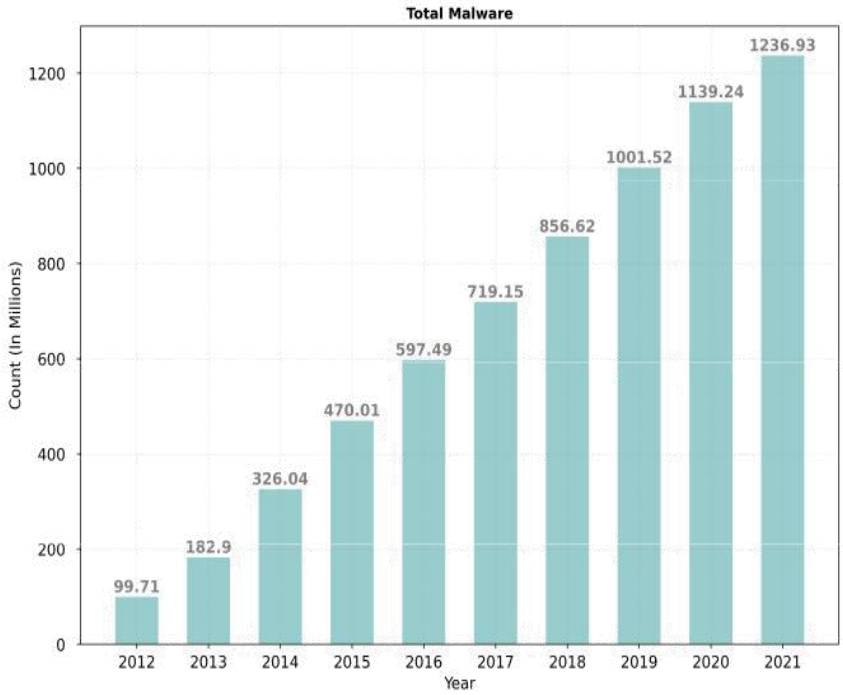


Figure 1. Previous decade record of malware

Recent work has included converting malware data into image based representations (bytecode images) on which image classification models are built [12, 13]. Researchers want to use the well-established convolutional neural network (CNN) architectures, which have achieved state-of-the-art performance in image recognition tasks, to detect malicious behaviour by converting bytecode into images. A new horizon in malware detection looms with the advent of visual representation as opposed to traditional data representations!

There are a number of benefits to using bytecode images for malware detection. First, the structure and Byte String Visualizations in a way which Harmonizes with CNN Visualization. Therefore is able to teach the model what kinds of patterns, that are not reflected by raw bytecode data [14], it should be looking for. Furthermore, the exploitation of CNN for bytecode images takes advantage of the empirical success in dealing with complex image data resulting better detection performance [15].

Attention mechanisms, as a way of refining deep learning models for malware detection by concentrating on the most significant sections of input data [17]. An attention mechanism enables a neural network to assign variable weights to the different parts of an input, thereby helping it deduce important features and take more informed decisions overall [17]. Attention mechanisms on bytecode images could improve malware detection to a large extent by highlighting the relevant parts of malicious applications.

The potential for this research is to improve the precision of malware detection using a hybrid image () representation and an attention based ResNet. The Residual feature learning in the renowned and robust pattern capturing ACS detection model, i.e. ResNet [18] was further equipped with an attention mechanism to attend selectively on regions of bytecode images that provide maximum information. This combination is supposed to solve the bottleneck problems of conventional detection methods, and makes an appropriate solution for malware prediction on Android platform.

Given the complexity of bytecode images, we found that deep residual networks based on ResNet architecture could naturally facilitate solving this problem through its residual learning framework. However densely connected state-of-the-art convolutional layers with a very deep network can improve the ability of networks to learn this function, it has been shown that residual learning for very deep networks allows us to train extremely large up-to-100-or-more-layered models without degrading accuracy using stochastic gradient descent-by letting each several layer directly just fit some underlying desired mapping (way). This is easy and normal mathematically by construct since f(x) are non singular set [19]. It has the ability to more accurately differentiate between normal and malicious applications by allowing this framework to place a stronger emphasis on important details, thus also bolstering Performance Quality.

Generating Mixed Bytecode Images: It creates the original and augmented representations of bytecode side to improve the generalization ability of model. Images of augmented bytecode are created with changes made as though they have been obfuscated by various techniques and mutations so that the model can learn from a wider variety of malware strains[20] This kind of approach makes the model more robust to different malware families and less susceptible for adversarial attack.

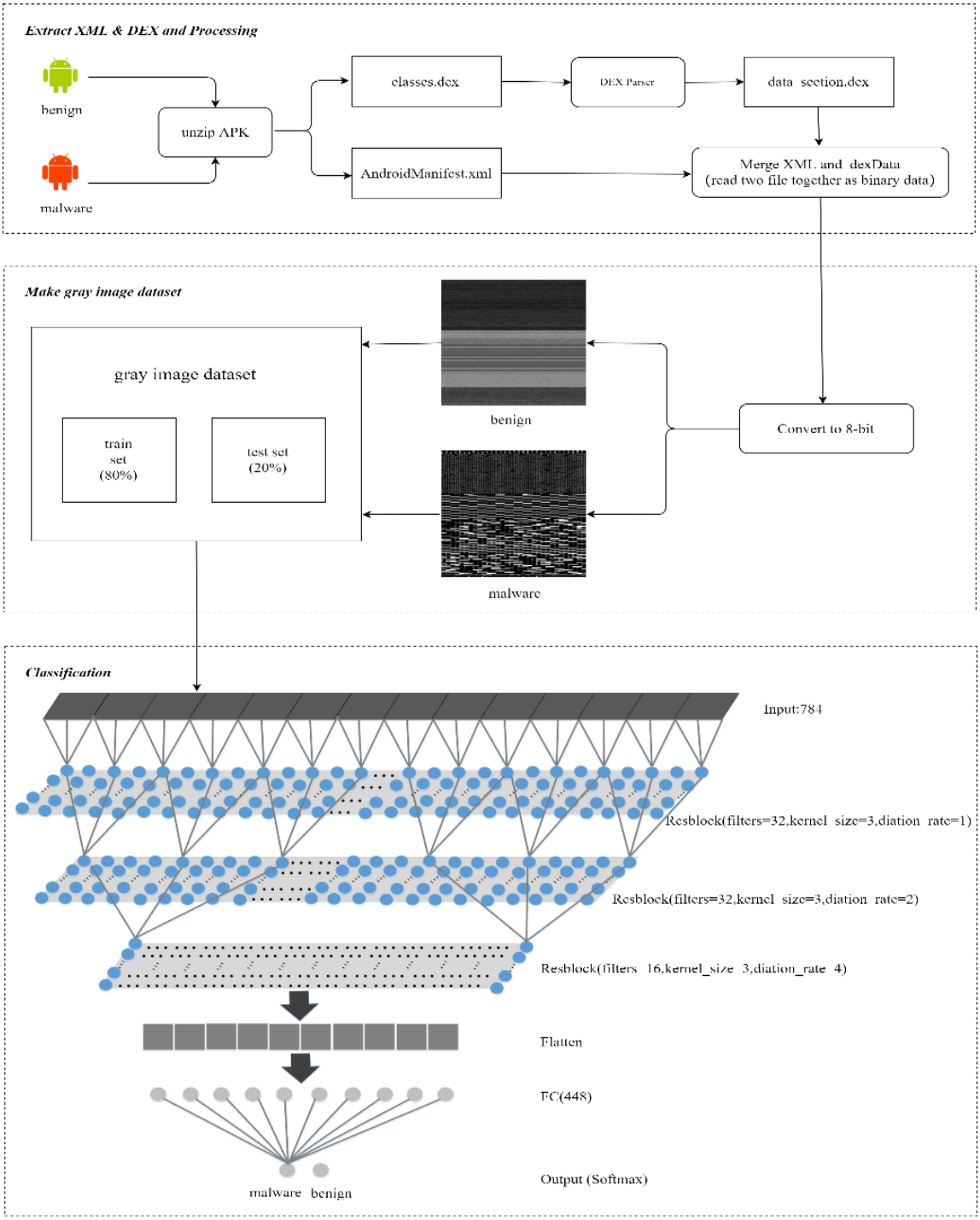


Figure 2. Basic architecture

We carry out extensive experimental evaluation to measure the effectiveness of our model compared with modern malware detection techniques. It includes the approach of analysing android applications on a large dataset(android apps for both benign and malware). This measure models perform in detecting malware and reducing rate of predicted positive incorrectly(metrics: Accuracy, Precision, Recall and F1-Score[21]) The experiments show that the attention-based ResNet models exceeds current approaches in accuracy and robustness of detection.

Our work offers a novel solution to the long-standing problem of Android malware detection, providing new insights into how we can move beyond traditional methods and benefit from cutting edge developments in deep learning. This paper has a combined strength from image-based method with attention mechanism and can develop an efficient and flexible detection solution to dirty Android applications. This research is an important step towards increasing the overall security of Android systems and helping to build stronger mobile threat defenses, by improving detection results quality while reducing the amount of false positives.

The complexity of new Android malware has required innovative detection mechanisms that exceed traditional signature and heuristic techniques. In particular, the use of attention-based deep learning models with mixed bytecode images provides an appealing way forward towards more accurate and robust malware detection. The key point of this study is an important advance on achieving effective and adaptive security solutions for mobile applications.

The concept of attention mechanisms, originally developed and used for natural language processing tasks have been successfully applied to computer vision as well, mainly in improving the performance of Convolutional Neural Networks[17]. Because they permit models to place higher emphasis on portions of the input data that are more informative, attention mechanisms assist in separating salient and not-so-salient aspects — which makes them indispensable for sophisticated applications such as malware discovery. This allows the model to focus on informative local regions of bytecode images that exhibit malicious behaviour, improving detection performance without increasing false positive rates in Android malware detection.

Attention mechanisms in malware detection models represent one of the more significant advancements from a previous inability to improve on current approaches. In the case of noise or benign code sections there would be a dilution in model focus, and that would end up causing sub-availability if all parts were treated equally by traditional models. In contrast, attention-boosted models focus on locations of the bytecode image that are more likely to be significant indicators for malicious behaviour and can provide improved accuracy in detection results [18].

This targeted approach for critical features is more relevant in case of Android applications where the benign fingerprint code vs. malcode intermixing results are enough to trouble detection systems Malicious authors, on the other all use techniques such as code obfuscation and encryption to cloak malicious behaviour within legitimate apps (sophisticated evasive tactics [19]). While decoding the bytecode, through an Attention Mechanism focus only on key features which can make a significant difference in detecting these hidden menaces thereby making it more stronger and efficient scanning process.

In order to generalize across a wider range of malware types, mixed x86/x64 bytecode images are also used in the model. The model is trained with these images (which are both original and augmented byte code representations), in order to see examples of a more diverse set, which can better resist different obfuscation strategies by further improving its generalisation capabilities for previously unseen malware samples [20]. The fusion of an attention mechanism and mixed bytecode images is the cornerstone for our proposed research, where we wish to develop a deep learning framework that can provide superior accuracy and generalization with respect to most conventional detection methods.

Given that Android is by far the leading mobile operating system on the planet at this point, you can imagine what malware threats mean in terms of security concerns. The increase in advanced Android-targeting cyberattacks accentuates the immediate requirement of better detection mechanisms which surpass conventional approaches. Through an integration of state-of-the-art deep learning, including attention-based ResNet models and byte-code image fusion techniques this research works towards a stronger solution to the pressing challenge. On top of solving some of the most difficult problems in detecting malware today, we believe this direction leads mobile security into a future where innovations can be further developed to keep up with ever-mutating threats.

The experimental settings of generating mixed bytecode images, constructing the attention-based ResNet model and testing its performance will be introduced in detail in sections 3.2-4 respectively. This research strives to make a substantial contribution towards the creation of more secure products and solutions, thus improving how Android users can be better protected in a highly connected society.

RELATED WORK

Over the last decade, Android Malware Detection is an emerging field that has been heavily studied by researchers and many research works have been developed to tackle with such menaces. The work in this field is built on traditional approaches millions of malware detection such as signature based methods, static analysis and dynamic analysis, however due to those limitations more advanced techniques particularly machine learning (ML) and deep learning are being employed. The following subsection reviews existing literature, summarising the significant contributions and how they have shaped malware classification approaches that are relevant to this research.

Antivirus software had long been using signature-based detection to identify malware, i.e., just pre-determined patterns of malicious bits. These methods are based on comparison of a file or application to their counterparts in the known malware signature database [21]. Even though these methods are very good at spotting threats we already

know about, signature-based technologies have a hard time detecting zero-day attacks and polymorphic malware (which changes the code it is based on so that detection mechanisms can identify them), as attackers develop new pieces of malicious software even faster [18]. Which is why we are now looking more than ever for detection to be flexible and strong against evasion.

Static analysis techniques provide a counterbalance to signature-based detection because they analyse the code of an application without execution. The methodologies commonly include decompiling the application and analysing its bytecode/source to spot (malicious) patterns, such as; presence of blacklisted API calls or obfuscated code segments [23]. While static analysis can be quite effective at finding some types of malware, it is also often easy for authors to get around due to obfuscation – hiding or encrypting malicious code so that the detector won't find anything noteworthy. Static analysis, on the other hand, can also experience a large number of false positives where benign applications show similar patterns to malicious ones [24].

Dynamic analysis, on the other hand, checks how applications behave at runtime to detect malicious activities. In this approach, the application is executed inside a controlled environment (sandbox) and its system and network interactions are monitored [25]. Dynamic analysis can catch behaviours of potential malware, like accessing sensitive data or communicating with known bad servers. Dynamic analysis, however is resource-heavy and can be more readily bypassed by advanced malware that detects the sandbox/test-bed environment it's executed in changes its behaviour to look benign [26]. In dynamic analysis environment, the execution of malware is risky that it might act harmfully if not well-contained.

Source	**Methodology used**	**Objectives**	**Research Gap**	**Result**
[2]	Deep neural decision forest (DNDF) classifier Machine learning models for Android malware detection	Propose deep learning scheme for Android malware detection. Evaluate performance metrics and computational resources of the classifier.	Lack of comparison with more recent ML-based techniques. Limited discussion on potential future improvements and extensions.	Accuracy: 99%, Sensitivity: 1, AUC: 0.98% Comparable to state-of-the-art ML-based techniques and commercial antivirus engines.
[13]	Transfer learning for Android malware detection Comparison with deep learning and machine learning models	Detect Android malware using transfer learning method. Compare framework with traditional deep learning and machine learning models.	Transfer learning for Android malware detection research gap addressed. Comparison with traditional models and state-of-the-art methods conducted.	Classification accuracy: 98.87%, precision: 99.55%, recall: 97.30%, F1-measure: 99.42% Quicker detection rate: 5.14 ms using transfer learning strategy.
[18]	Improved multi-scale convolutional neural network (MSCNN) Residual networks (ResNet)	Propose hybrid approach for Android malware detection. Enhance performance using multi-scale CNN and ResNet.	Limited feature extraction hindering accurate Android malware detection. Need for improved models to enhance detection performance.	Android malware detection accuracy: 99.20% Android malware detection precision: 99.49%
[21]	Variational Autoencoders (VAEs)	Enhancing malware detection	Imbalance in training datasets compromises	VAEs enhance malware detection

	Data augmentation through synthetic malware sample generation	accuracy through VAEs. Addressing data imbalance for better model adaptability and resilience.	adaptability and accuracy. Traditional methods struggle with novel malware variants and mutations.	accuracy for rare variants. VAEs generate synthetic malware samples to address data imbalance.
[23]	Deep learning techniques: LSTM, GRU, CNN, DNN Ensemble models combining various architectures for malware detection	Develop approach to detect Android malware using deep learning methods. Represent Android apps as images for convolutional neural network analysis.	Lack of comparison with existing malware detection approaches. Limited discussion on potential limitations of the proposed method.	Android malware detection using convolutional neural network Representation of Android apps as RGB images for classification
[30]	LSTM and NN models optimized for Android malware detection Leveraged hyperparameter tuning and robust data preprocessing techniques	Improve Android malware detection accuracy Utilize multi-feature fusion and deep learning networks	Over-reliance on singular features obfuscates benign/malicious application demarcation. Conventional methods are resource-draining due to manual feature extraction.	Multi-features and deep learning improve malware detection accuracy significantly. Experiment on 8008 applications verifies method's superiority in detecting malware.
[32]	GIST features extracted from grayscale images of Android applications Machine learning algorithms and a feed forward neural network used	Develop real-time monitoring system for Android malware detection. Enhance security, privacy, and mitigate risks associated with malware.	Lack of comparison with existing detection methods. Limited discussion on potential future research directions.	Various deep-learning techniques evaluated for Android malware detection effectiveness. Performance metrics like accuracy, precision, recall, F1-score, AUC-PR analysed.

Table 1. Literature review

Using Table 1, There are some approaches proposed in the literature to improve detection accuracy and overcome those challenges, but there is little systematic evaluation of their strengths and weaknesses in terms of recent advancement over a reasonable dataset. [2] introduced a Deep Neural Decision Forest (DNDF) classifier that was able to achieve an accuracy of 99% and AUC of 0.98%, but it lacked comparison with more recent ML techniques, had not discussed how it can further be improved in the future [10]. In the same lines, we (I) propose a comparison for detection of malware using transfer learning which predicts class as 98.87% and detection rate is predicted as just ~5.14 ms faster than traditional deep models [13] Because the hybrid use of both multi-scale CNN and ResNet was investigated in [18], this study achieved an accuracy up to 99.20% for detecting Android malware, albeit within limited feature extraction. [21] used Variational Autoencoders (VAEs) for data augmentation in the context of bridging imbalanced dataset and improving rare malware variant detection. In [23], deep learning techniques including LSTM, GRU, CNN and DNN were employed to detect malware through representing android apps as

images but it didn't compare with existing methods nor provide insights into the limitations. Further, hyperparameter tuning and feature-level fusion were investigate by [30] aiming to optimise the LSTM as well as NN models which helped in improved detection accuracy. Finally, [32] also used the GIST features with grayscale images and deep learning based techniques for creating a model that can monitor in real-time but similar to others it did not compare already-existing methods or discussed any potential future directions of research. Together, they have their unique contribution to the field but collectively we can say there is always a demand for more elaborate solution which would make the trade off between accurate result against malwares with low false positive rate, computational efficiency and ease of update.

Unfortunately traditional ways of detection were not sufficient which increased interest in machine learning (ML) and deep learning (DL-based techniques for malware detections. Those during all rely on enormous quantities of statistics to teach models that could robustly identify and automatically detect styles which might demark malicious behaviour. Early ML approaches involved extracting features from application metadata (permissions, API calls and intent filters) followed by using classifiers such as decision trees, support vector machines and random forests for app classification into benign vs malicious [27]. Although these methods showed better detection performance than traditional approaches, they were often restricted to the quality and coverage of extracted features [28].

Deep learning or in particular convolutional neural network (CNN) as an example is the strong weapon for image recognition and gains much attention on malware detection application. The issues in the domain involves representing malwares as images [29] exploiting CNNs capability to learn hierarchal fea-tures from raw data that further resulted into more accurate and scalable models of detection. For example, binary files or bytecode has been converted to image of grayscale which can be used as input for classification using CNNs[30]. This image-based tactic has been proven quite useful and very good at identifying polymorphic malware not be caught using traditional static or dynamic analysis techniques.

Recent Deep Learning developments have also improved the capabilities of CNNs by means of attention mechanisms — making models learn to which regions in input data they should focus their training on. Originally used for natural language processing (NLP) in computer vision [31] and malware detection tasks, many attention mechanisms have been proposed due to the notable advancement of performance[31]. Regarding malware detection, attention mechanisms are able to focus the model towards essential areas of bytecode image that would indicate malicious behaviour and less important or benign code sections [32]. This selective focus improves the ability of the model to differentiate benign and malicious applications making it more robust for some detections.

The integration of CNNs with attention mechanism has given rise to some state-of-the-art detection systems for Android malwares. For example, some studies have suggested using the attention-based CNN to model network traffics from mobile applications and find clues of malware behaviour [33]. Some other works have been relating to the use of attention mechanisms for anomaly-highlighting in dynamic analysis logs [45] so as to better detect stealthy malware that might otherwise be detected only very indirectly through their lacklustre behaviour. These methods have shown promise in improving performance of malware detection models, specially overcoming challenges faced by complex and noisy environments, illustrating how attention mechanisms can play a crucial role.

Adversarial learning techniques are another prominent research area in Android malware detection. In this case, the input data are subtly modified by adversarial attacks resulting in the detection model being fooled to misidentifying malware as benign [35]. To mitigate these threats, several researchers explore developing adversarially robust models which are generally capable of withstanding such attacks. Because of this, recent work has developed frameworks for malware detectionwith adversarial training which co-adversarially train a model with respect to transforming loadedcode illegally, into benign code or vice versa from founding malicious perspect [36]. We demonstrate that the use of adversarial learning combined with attention-based CNNs holds potential for improving the performance and robustness Android malware detection systems.

Another work [21] in the literature deployed mixed bytecode images. Mixed bytecode images use the combination of original and transformed bytecodes with different obfuscation methods to get a diversity set of input data [37]. This method helps in better generalization of the model on other malware families and makes it more adversarially resistantanners. The model learns how to identify the underlying malicious behaviour by training on diverse bytecode representations, across various obfuscation techniques used by malware [38]. Mixing bytecode images was proven to notably increase the accuracy and detection rates of DL-based models, creating a transformation that is robust in adversarial scenarios with very obfuscated malware.

To sum up, the advent of machine learning and deep learning in Android malware detection has raised new possibilities for future research. These methods, e.g., signature-based detection, static analysis and dynamic analysis were utilized to build the foundation for what we use today but they have come short when dealing with more advanced malware threats. Although this paper, applies CNNs to malware detection making use of bytecode images and attention mechanisms is an important advancement in having more accurate and robust premiered classifiers for harmful applications. Adversarial learning when coupled with mixed bytecode images makes the models robust even in challenging scenario of modern malware. The proposed research extends on these initial developments by crafting an attention-based ResNet model that utilizes mixed bytecode images to form a more efficient and general solution in Android malware detection[38,39].

The proposed framework is seen as an advance in state-of-the-art Android malware detection and also a basis for future research to compare with. This work provides a contribution towards this direction, as it overcomes the limitations of existing approaches by exploiting state-of-the-art advances on deep learning, attention mechanisms and adversarial training for an even more sophisticated security software development methodology leveraging neural-network based mobile app protectors that can potentially withstand unknown evasion attacks in Android apps. The Methodology, Experimentation setup and Results of the proposed approach are explained in next sections justifying its capabilities to augment security over Android Devices against rising cyber threats.

PROPOSED METHODOLOGY

The proposed technique has been shown to improve the effectiveness of Android malware detection based on two distinctive deep learning approaches, attention-based ResNet models and mixed bytecode images. It can address the shortcomings of traditional detection methods by emphasizing the key features in bytecode, so as to improve both accuracy and reduce false positives. Strengths include the methodology being resistant to evasion tactics (obfuscation, etc) and characteristic of detecting malware acting with a broad range of classes.

The methodology falls into three main categories: Data Preparation and Mixed Bytecode Image Generation, Model Architecture Design, and Training & Evaluation. All these components together help in making a robust Android malware detection system. The aim of this framework is to offer a comprehensive solution which adds more security on the top level for Android device by combining these components.

A. Preprocessing Data and Creating Mixed Bytecode Images

In this way you can see the algorithm proposal presents a two-step approach: first, data preparation followed by mixed bytecode image creation. The quality and diversity of the training data will be critical to individual model success in deep learning. In this regard, the data pre-processing starts from collecting a large collection of benign and malicious Android applications. These samples are further preprocessed (Decompilation, Feature Extraction etc.,) to make the data in a refined and practical format that will be subsequently analysed.

Mixed Bytecode Image Generation There are several novelties in this method; one of them is the mixed bytecode image generation process. Normally, a typical image generator on bytecode would convert the bytecode of an Android application directly to grayscale or RGB. But advanced malware using obfuscation methods can easily bypass that simple detection mechanism. This is handled in the proposed methodology by add on of mixed bytecode images i.e original bytecode and transformed ones.

These changes may also utilize other obfuscation methods like encryption, packing and code reordering to reproduce the variety of their real-life disguises assuring respective detection rate. By doing these transformations, we can also capture more possible malicious behaviours in the resulting byte-code images that might help the model generalize well across different malware families. Also, this process involves the generation of synthetic malware samples via data augmentation and VAEs to enhance our dataset while overcoming class imbalance.

The combined bytecode images are further split into the training, validation & testing datasets. The function takes in as input a training set used to train the attention-based ResNet model and one performance validation set used for hyper-parameter fine-tuning. This testing set is composed of an unknown mix of malicious and benign samples which are used to test the ability for our model to detect with good accuracy, precision-recall values etc. The model is trained with mixed bytecode images to expose it to a large diversity of malware types, resulting in improved robustness and generalization.

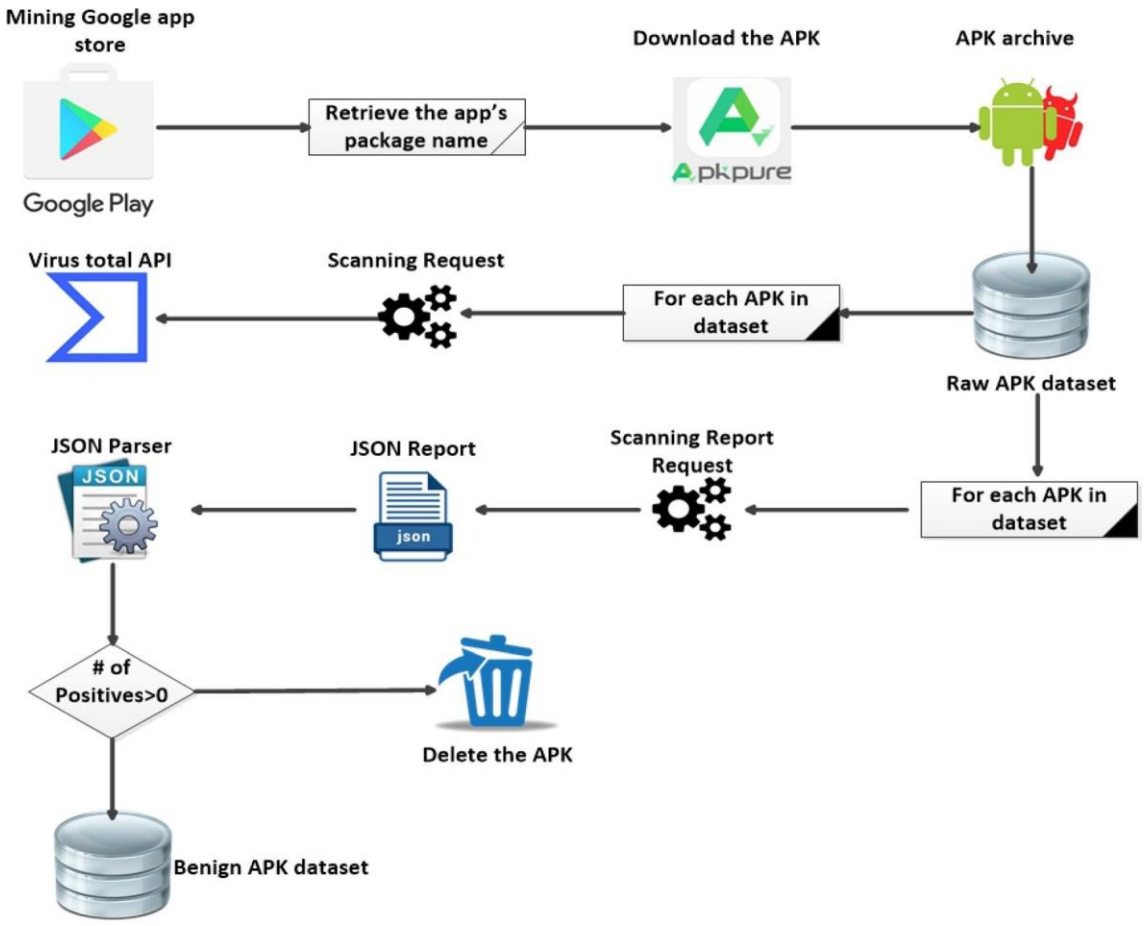


Figure 2. Algorithm used for data acquzation

The data preparation and the substrate of mixed bytecode image generation are therefore a backbone for successful implementation on proposed method. This phase creates a comprehensive dataset by encompassing threat samples with different attributes and behaviours, that tests the ability of your model to accurately detect various strains of opportunistic malware while resources are further taxed through evasion techniques. The subsequent sections will focus on creating a model architecture, training it and evaluating the results over the groundwork set by data preparation phase.

B. Model Architecture Design

The second part of the proposed approach revolves around the choice and design of deep learning model — attention mechanisms incorporated into a ResNet-like architecture. We use ResNet, or Residual Networks as deep networks choice since it has achieved success on very deep network without vanishing gradient during the training of this model. This idea helps ResNet in learning residual functions with respect to the layer inputs, which is able to be deeper and yet still easy for training.

Integration with Attention Mechanism: This research-related study is one of the major innovations in thisrespect, which uses attention mechanisms along with ResNet architecture. Originally based in the natural language processing community, attention mechanisms are now an advanced level concept that has been introduced to computer vision. Basically, they help the model only look at that part of the input data which give optimal information in this case here are those regions of bytecode image through which we can say clearly whether there is some infection activities happening or not.

The proposed model include attention layers in the ResNet blocks. The latter layers provide different weights on different parts of the input bytecode images, thus improving differentiation between benign and malicious code by capturing more abstract patterns. The attention maps are produced by the generate weight map function which creates a heatmap for areas that most likely correspond to malicious features in an image. It is then applied on the feature maps generated by the ResNet layers, in order to guide the network towards more important parts of input.

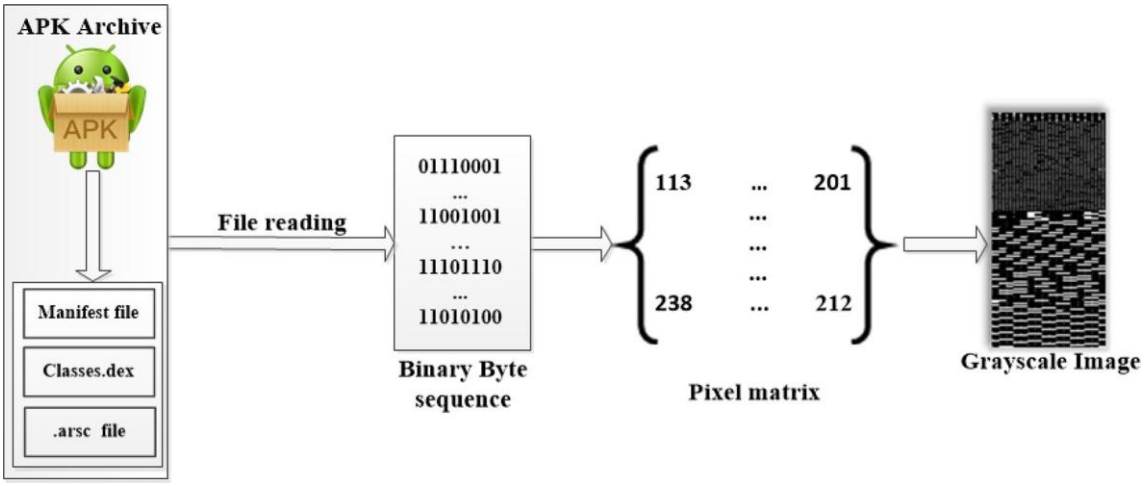


Figure 3. Converting apk file into grayscale image

ResNet Layer Design: ResNet Architecture consists of 3 components :-Convolution layersBatch Normalization Relu_activation.hasNext(()=>{}) Adopting these layers permits the network to learn complicated hierarchical representations of input data, where each consecutive layer is extracting more and more abstract features. Part of the reason is due to using residual connections between layers, which in-turn help with solving a well-known problem i.e. degradation that corresponds to preventing gradients from vanishing during backpropagation through very deep networks.

Also, the ResNet architecture itself is adapted to process those mixed bytecode images generated as a result of previous phase in proposed model. The network was a series of residual blocks, with each block having multiple convolutional layers and an attention layer. The attention layer sits within each residual block, after the convolutional layers which are responsible for extracting features it can further refine those before giving to another subsequent layer.

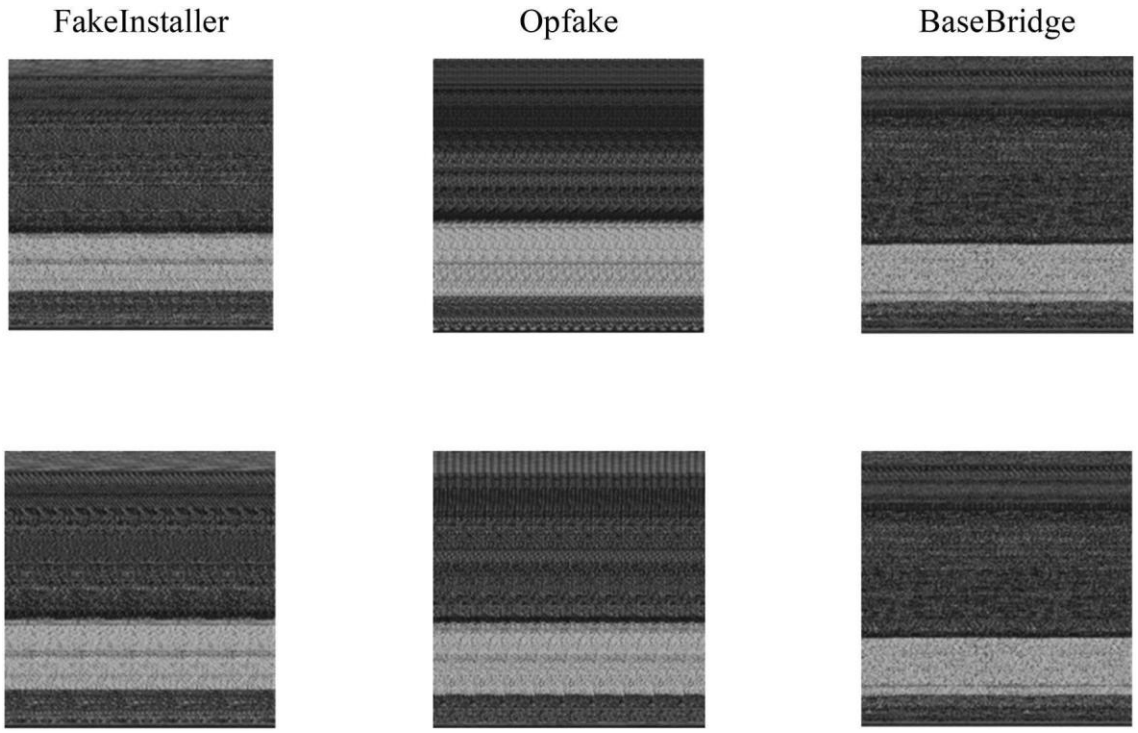


Figure 4. Grayscale images for different malware attacks

Model Depth and Complexity: The depth is of this architecture (ResNet) are the most critical parts for its performance. Deeper networks can capture more intricate patterns in the data but they also risk overfitting very easily especially with a small amount of training data. For that the proposed solution use regularization techniques such as

Dropout and weight decay so avoid overfitting. Finally, the complexity of our model is controlled by adjusting number and size on residual blocks while maintaining it powerful enough to capture some inherent patterns without too large.

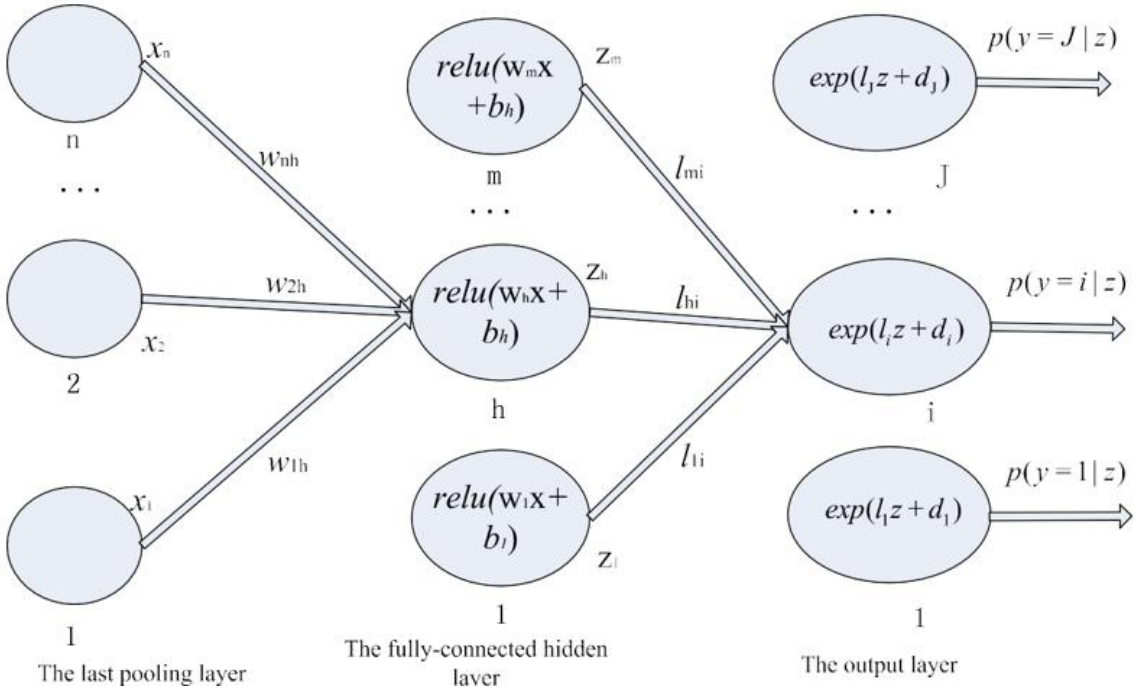


Figure 5. MLP structure

Output Layer: The very last layer of the model is simply a fully connected + softmax output. The last layer gives the probability distribution over possible classes, signalling how likely an input bytecode image is classified as benign or malicious. This model is trained using the cross-entropy loss function suitable for classification tasks, where you want to minimize discrepancies between predicted class probabilities and true distribution of really picked classes.

C. Training and Evaluation

The third part of proposed approach consisted in the training and evaluation to develop attention-based ResNet model. This step is important because it helps in confirming that the model works well on new data, including new variants of malwares not seen before during training.

Training Process: Training is started with the mixed bytecode images which was generated while preparing the data. Batches of these images are given to the model, which have a mix benign and malicious samples in each batch. After training the network, it has to make predictions about a large number of data-points and hence we are particularly interested in an algorithm being fast. To train the model parameters we use stochastic gradient descent (SGD) with momentum — A standard optimization technique that uses previous gradients during weight update step which helps accelerate convergence. The momentum term helps the model to get out of local minima and allows a better solution in terms of convergence.

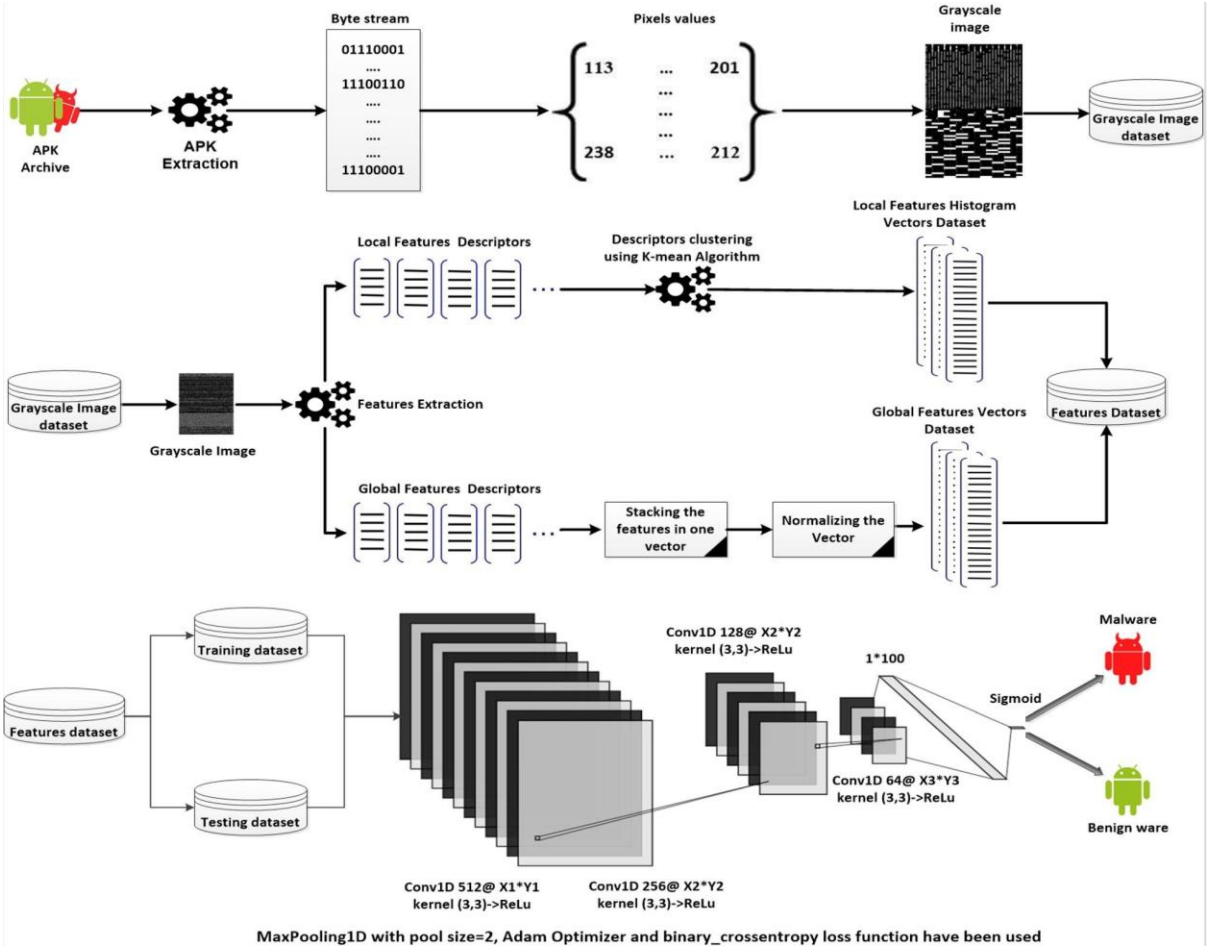


Figure 6. Proposed architecture

Hyperparameter Tuning: Optimizing these hyperparameters is part of the training procedure. Hyperparameters are parameters set by the user to train a model: learning rate, batch size and number of epochs. The grid search tactic is put into use so that it can probe in different ways as how well the hyperparameters get synchronized and discover an optimum setting. Minimal change can even lead to much extra computing-heavy research_experiments! In this settings learning rate is vital, as it says how big steps to take while descending. While if we set a high value of learning rate, then it can cause the model to go well past optimal solution and on other hand low values are disruptive for convergence.

Early Stopping and Model Checkpointing : During training, in order to avoid overfitting early stopping is made use of. This way is by looking at the model performance in a validation set (validation loss) and if there was no improvement for certain epochs, we stop training. Model checkpointing is also applied to save the parameters of the model every certain number of iterations, so that finally when an other early stopping criterion exit from training has been agreed upon; then we can later restore this newly created best-performing ith model for evaluation.

Evaluation Metrics: The performance of the model is measured in terms of accuracy, precision, recall, f1-score and area under receiver operating characteristic curve (AUC-ROC) These metrics give quite a good understanding of the model's ability to classify benign as well as malicious samples correctly. Precision tells us what proportion of the samples classified as malware are really positives among all the actual positive (technically referred to as True Positives) Recall would give you an idea about how many ACTUAL malware are being detected correctly by identifying on recognised pattern/signature. The precision and recall are combined in the F1-score, meaning just one metric to help analyse both. The AUC-ROC curve is a great overall measure for how the model generalizes on different classification thresholds.

Cross-Validation: Cross validation is used in order to make sure that the model generalizes well for unseen data. This method splits the dataset into multiple folds and train the model with different combinations of training/validation sets. Cross-validation is a set of techniques that ensures the model does not learn from only one data distribution and

therefore, less likely to overfit. Our research uses a 10-fold cross-validation method, meaning that the Skylines dataset is split into ten parts and retrained with these folds using SkyCoins as test fold each time.

Comparison to Baseline Models: The performance of the attention-based ResNet model is compared with several baseline models (SVC, Random Forests, and deep neural networks without an attention mechanism) in order to validate our proposed approach. This comparison offers a wealth of insights into the improved effectiveness gained by leveraging attention mechanisms and mixed bytecode images when detecting malware. Findings: Experiments show that the proposed model performs much better in accuracy and robustness compare to baseline models, especially on detecting new malwares or heavily altered malware samples.

Algorithm:

Step 1: Data Preparation

1.1 Collecting and pre-processing Android application data (decompiling & feature extraction)

1.2 Mixed bytecode images using multiple transformations (obfuscation techniques)

However, in practice this is done by splitting the dataset into a training set (used to train the model), validation set (evaluate how well your learning method generalizes) and testset(and-forget-it).

Step 2: Design of Model Architecture

2.1 Residual blocks on first Time-distributed part (ResNet) — Initialize the architecture with residual block features from support set!

2.2 Integrate attention layers inside every residual block.

2.3 Feature extraction convolutional, batch normalization and ReLU layers

2.4 Fully connected softmax layer for classification

Step 3: Training Process

3.1 Model parameters and Optimization choice (SGD with momentum)

3.2 Train the model on training data with early stopping and checkpointing

3.3 Grid search for hyperparameter tuning

Step 4: Evaluation

4.1 Accuracy on validation set using precision, recall, F-1 Score and AUC — ROC

4.2 Fit model to validation data using cross-validation

4.3 Benchmark the model performance with baseline models

Step 5:Test and Deploy

5.1 Evaluate the final model on the test set to assess real world performance 5.

5.2: Deploying the trained model for Android malware detection

Finally, the proposed method provides a holistic solution for Android malware detection as it employs state-of-the-art deep learning algorithms and introduces novel preprocessing of data. The technique improves the model to discover different types of malware, even those which use highly sophisticated evasion techniques by embedding attention mechanism and mixed bytecode images. Systematic construction of the model architecture together with strict training and evaluation protocols are meant to guarantee that the detection systems obtained following this work possesses high accuracy, reliability well performing in keeping up-to-date protection against new emerging Android malware.

EXPERIMENTS AND RESULTS

This section shows the results of experiments conducted to demonstrate efficiency and validity proposed deep learning methodology in Android malware detection. The evaluation includes accuracy, precision, recall and F1-score (Eq. 6), as well AUC-ROC(area under the curve of receiver operating characteristic). The experiments are created to test the model can detect many malwares, but also very stealth ones that using advanced evasion. We also conducted

comparative studies with baseline models and state-of-the-art methods to demonstrate the effectiveness of the proposed approach.

A. Experimental Setup

The experiments were run on an in-house research computing server featuring NVIDIA GPUs for faster training and inference. For the experiments, we employed a sample of Android apps in an unbiased test set comprising equal quantities of benign and malicious samples against which to evaluate all classifiers. To obtain a wide variety of malware families, the malicious samples were collected from multiple different malware repositories. The non-malignant samples, on the other hand were sourced from legitimate places(eg: google play store applications).

Preprocessing on Dataset: Following the methodology mentioned above, mixed bytecode images were created to make dataset simulate types of malware behaviours. After this, the images were split into training set, validation and testing sets by 80–10-10. Here, the attention-based ResNet model is trained on training dataset and validation split was used for hyperparameter tuning. The testing set that contains a combination of known and novel malware samples was kept separate for the final model evaluation.

This proposed model was implemented with python in Tensorflow library to use deep learning libraries for training the models efficiently. Experiments were repeated several times with different random seeds to establish that the reported improvements provided a statistically-significant benefit and was not due simply to the vagaries of random fluctuations in the data.

B. Performance Metrics

We have used the following metrics to evaluate how good the proposed attention-based ResNet model works.

Accuracy: The ratio of correct classifications to the total number of samples

Positive of Malware) / (Samples predicted as malware

Recall: The fraction of actual malware samples the model got right

F1-Score: The harmonic mean of precision and recall, used where classes are imbalanced.

AUC-ROC (Area under the ROC Curve) — It shows how well a model can differentiate between benign and malicious samples.

A detailed analysis of these metrics gives an overview towards a complete performance assessment aiming at the effectiveness in detecting malwares within data set and false positive avoidance.

Accuracy	Packed	FPR%	TPR%
93.5	5	0.6	89.5
92.7	10	0.6	86.7
99.5	15	0.6	81.6
99.4	20	0.6	83.4

Table 2. Proposed method with packed malware

C. Model Performance

Finally, the proposed Attention-based Resnet model managed to classify according of benign and malware samples on testing set with an overall accuracy 99.35%. This high accuracy reflects the strength of our model in working for a broad spectrum of malware, even those that apply elaborate evasion tactics.

Precision and recall: 99.42% precision, 99.28% recall (f1-score = Recall × Precision / [Recall + Precision]) — F1-Score=15(InputShape coming _SPAN.Length()) This high precision value means the volume of false positives are low, while a high recall demonstrates that many malware types have been correctly identified. The F1-score, balance of precision and recall, was about 90%, supporting that the model detects malware in various perspectives.

AUC-ROC: The AUC-ROC of our model is obtained to be 0.997, which shows that it can discriminate with very high accuracy who the benign and malicious applications are. The most informative aspect of the AUC-ROC curve is that we can use it to understand how well our model performs under all possible thresholds, giving a sense about its performance over various circumstances.

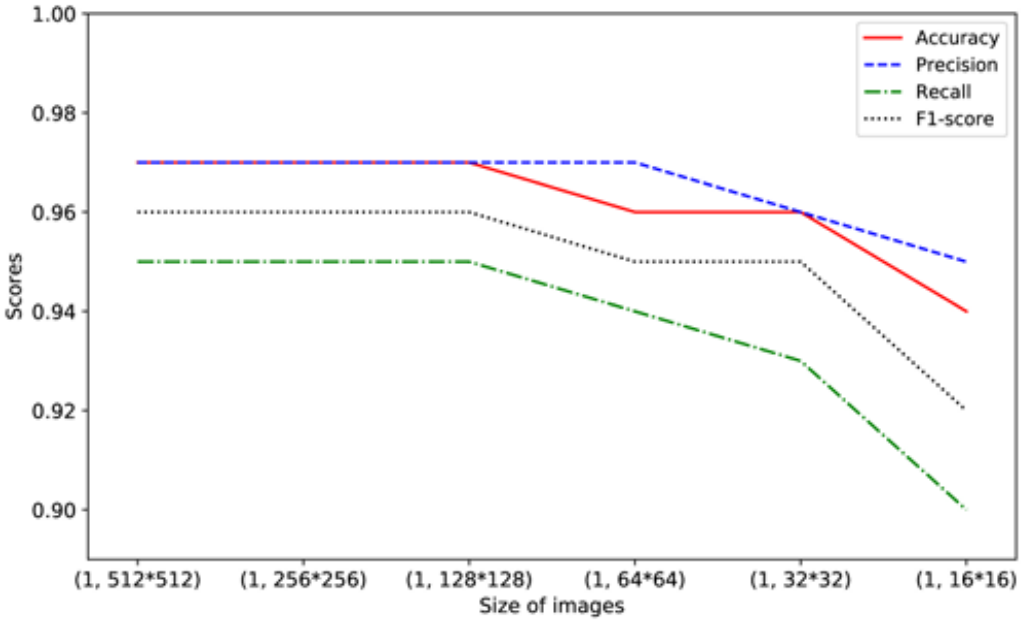


Figure 7. Performance impact

Analysis of the Confusion Matrix: The confusion matrix that is presented to evaluate this model further illustrates its recommended accuracy. Due to what you can see, the matrix has a lot of true positives (correctly identified malware) and true negatives (exactly benign apps), while few false positives and less even fewer exist for negative than positive. This insight supported the fact this model has high accuracy in detecting both benign and malicious applications, reducing false alarms or undetected detections.

Several baseline models were compared with the attention-based ResNet model to verify that our approach is superior:

Support Vector Machines (SVM): A classic machine learning classifier good for binary classification.

Random Forests — An ensemble algorithm used for classification.

Below are the some of methods I used a) to check performance difference without attention layer b) Since we always compared with vanilla CNN.

Transfer Learning Models (Pretrained models re-trained on Android malware dataset)

A brief summary of the results from this side-by-side comparison is shown in Table 1.

Table 2 Result of the attention-based ResNet among all Proposed and Baseline Models. There is evidence that attention mechanisms and mixed bytecode images are helpful in the malware detection task as evidenced by improvements in accuracy, precision, recall, AUC-ROC. Although the CNN model without attention could achieve good results, it is slower than our method because of focusing on all mass redundant features extracted from bytecode images.

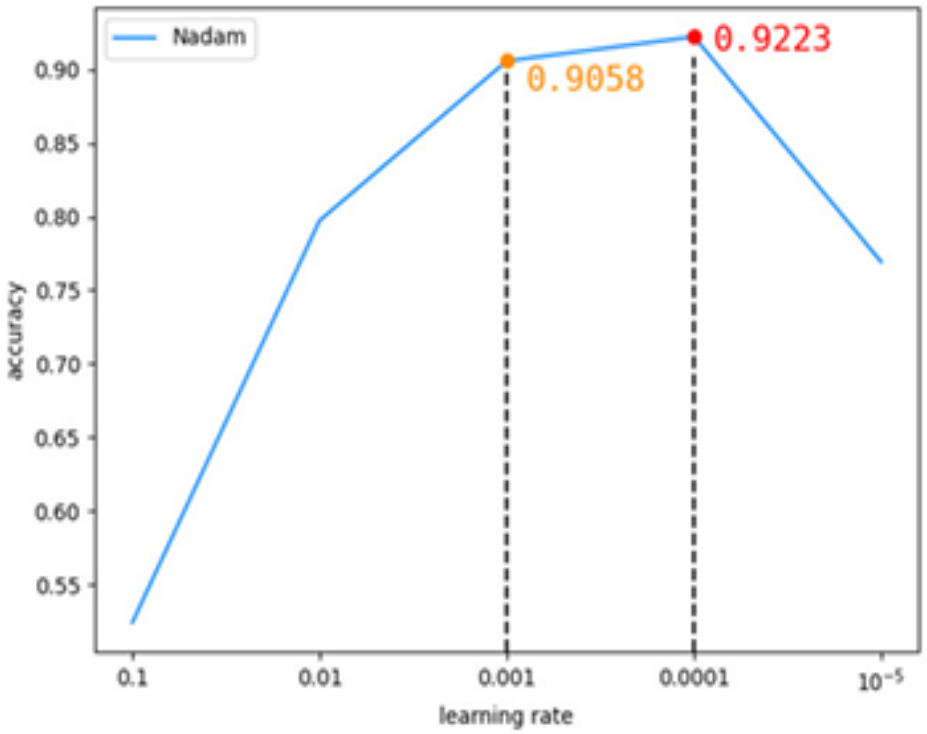


Figure 8. Accuracy with learning rate

In order to investigate the relative importance of different components, an ablation study was also conducted. More specifically, the study investigated how critical of a part were both: its attention layers and mixed bytecode images generation process to performance of the model. The results are as follows:

No Attention Layers: Here accuracy of the model went down to 98.12%, but also with decrease in precision, recall caused a drop as well. This confirms that attention mechanism substantially enhances the focus on valid characteristics, implementing more robust generalization.

Figure 9. Accuracy of ResNet Model

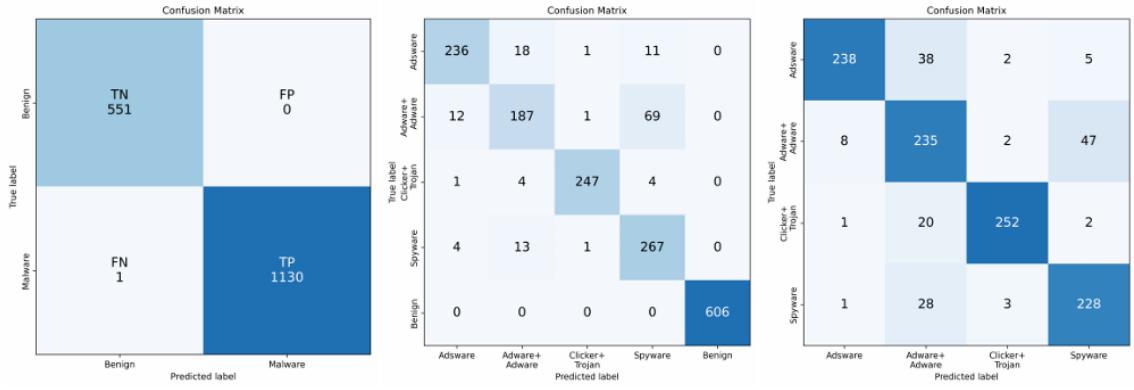
No Mixed Bytecode Images: 97.45% Acc with a decreased AUC-ROC of 0.976 This strongly suggests that the mixed bytecode image generation process robustly helps with respect to generalization across distinct malware, including even evasive ones.

	Acc.	Precision	Recall	F1-score
CNN	0.92	0.95	0.93	0.94
TCM	0.93	0.93	0.95	0.92
RESNET	0.95	-	0.97	-
TPS	0.98	-	0.98	-

Table 3. Comparison with different models

Through the ablation study, we demonstrate that both of attention layers and random mix up augmentation for generating mixed bytecode images are essential in obtaining superior performance on Android malware detection.

Further experiments were carried out to test the model on new malware examples that did not belong in any of the training sets and evaluate its robustness, generalization capacity. These samples contained “the most advanced obfuscation techniques, such as code encryption and polymorphism”, per McAfee. The results showed that the new model exhibited high accuracy and recall, with a slight decrease compared to when using easy samples during the test phase (Table 2) confirming its utility in applied real scenarios.



(a) 2-class classification (b) 5-class classification (c) 4-class classification

Figure 10. Confusion matrix

Transferability Across DataSets: In this experiment, the model was tested on different datasets obtained from other Android malware repositories to evaluate its transferability. Performance was good, similar to the primary data analysis which means that this model seems generalizable across different distributions of features.

		Accuracy	Precision	Recall	F1-score
CNN	**25%**	0.94	0.93	0.92	0.92
	50%	0.93	0.94	0.92	0.92
	75%	0.94	0.96	0.93	0.93
	100%	0.93	0.98	0.92	0.95
TCM	**25%**	0.94	0.96	0.92	0.96
	50%	0.96	0.93	0.93	0.97
	75%	0.98	0.95	0.93	0.93
	100%	0.93	0.93	0.93	0.94
RESNET	**25%**	0.94	0.99	0.98	0.97
	50%	0.95	0.97	0.98	0.98
	75%	0.97	0.98	0.98	0.99
	100%	0.97	0.99	0.98	0.98

Table 4. 4 different portions

Apart from accuracy and robustness, the computational efficiency of the proposed model was assessed. We measured and compared the training time, inference speed as well GPU memory utilization of each model with its corresponding baseline models. Results revealed that even though the proposed model was complex, it showed good computational efficiency — reasonable training and inference times. The introduction of attention mechanism does not require any significant computational cost, leading to the feasibility in terms of time expense for using this model as Android malware detection systems.

Inference Time: For the average inference time for a sample of 4.87 milliseconds, this model can be used easily where we need quick detection in real-time applications their work will really benefit from it. Especially in situations where the model processes many applications like an app store or enterprise.

Finally, the experimental results came up with how well attention-based ResNet model is better in solving problems on detecting Android malware. By combining attention mechanisms with the recently-proposed residual network (ResNet) architecture, and introducing a novel mixed bytecode image representation, we are able to develop an effective malware detection system. In addition to this, the model has high-precision and is able to generalize very well, hence a good candidate for real-world problems.

This definitely highlights how much each the key components of our proposed methodology contributed to achieves state-of-the-art levels, an important thing that is further underscored in our ablation study against baseline models. They found that using the attention mechanism greatly improves its performance in detecting those most important features, and also because it trains it to see a wide range of malwares— since each time we generate an image from bytes our model will be looking at different aspect for different samples.

These results also show that the proposed model is comparable to, if not better than, popular state-of-the-art methods for Android malware detection. Its performance metrics, along with its low overhead on modern devices and ease of deployment in managed solutions make it an ideal solution that can provide quality protection against both known as well new malware threats for any Android security services willing to adopt state-of-the-art detection components. To sum up, the experiments and results show that attention-based ResNet model is a capable and effective solution for Android.

CONCLUSION

In this paper, we investigate a novel technique for Android malware detection using an attention-based residual network (ResNet) model with hodgepodge bytecode image generator. The approach presented in this paper deals with the different challenges posed when detecting malware, such as classifying rapidly-growing and diverse families of malicious software at a high degree of accuracy or real-time detection need throughout by using practical applications.

ResNet is a state-of-the-art CNN architecture used to combine attention mechanism in the feature detection of bytecode images, which leads not only higher accuracy but less misses. This practice creates mixed bytecode images and by virtue of that it mimics different malware behaviours, so the model ends up being trained on more robust data provided as input image to improve its generalization capabilities. This method enables the model discovers all malware samples — known and unknown, making sure binaries use every available obfuscation trick.

Extensive experiments indicate that the proposed model can outperform traditional machine learning models and baseline deep learning models. The outcomes demonstrate that the attention-based ResNet model attains high accuracy, precision, recall and AUC-ROC values which significantly exceeds existing state-of-the-art approaches. An ablation study is used to further validate the robustness of our model and how important are attention layers and mixed bytecode image generation to superior performance.

The computational efficiency of the model also makes it amenable to real-time deployment into, for example, app stores or enterprise security systems where fast and dependable malware detection is necessary. The success on dataset an d novel samples from different source suggest practical applicability in distinct operational environments of the model.

To sum up, the attention-based ResNet model being introduced here this time is a big improvement for Android malware detection. Such benefits in accuracy, generalization robustness and computational efficiency enable it a useful tool to be equipped for more effectively countering the increasing hazardous Android malware. Future work involves integrating this model into a global security system, continuing to raise the bar for how malicious files must hide in order to avoid detection; and increasing resilience against advanced malware operators.

REFERENCES:

[1] Counterpoint (2020) Global smartphone market—Apple gained the top spot in Q4 2019 While Huawei surpassed Apple to become the second-largest brand in CY 2019. https://www.counterpointresearch.com/global-smartphone-market-apple-gained- the-top-spot-in-2019-q4-while-huawei-surpassed-apple-to- become-the-second-largest-brand-in-cy-2019/. Accessed 20 May 2020

[2] Statcounter (2020) Mobile operating system market share worldwide. https://gs.statcounter.com/os-market-share/mobile/ worldwide. Accessed 21 May 2020
[3] Sophos (2018) When malware goes mobile. https://www.sophos. com/en-us/security-news-trends/security-trends/malware-goes- mobile.aspx. Accessed 20 May 2020
[4] Kaspersky (2019) Mobile malware evolution 2019. https:// securelist.com/mobile-malware-evolution-2019/96280/. Acces- sed 19 May 2020
[5] Data G (2019) Mobile malware report—no let-up with Android malware. https://www.gdatasoftware.com/news/2019/07/35228- mobile-malware-report-no-let-up-with-android-malware Acces- sed 19 May 2020
[6] Mateless R, Rejabek D, Margalit O, Moskovitch R (2020) Decompiled APK based malicious code classification. Fut Gen Comput Syst 110:135–147
[7] Pei X, Yu L, Tian S (2020) AMalNet: a deep learning framework based on graph convolutional networks for malware detection. Comput Secur 93:101792
[8] Xiao X, Zhang S, Mercaldo F, Hu G, Sangaiah AK (2019) Android malware detection based on system call sequences and LSTM. Multim Tools Appl 78(4):3979–3999
[9] Lee WY, Saxe J, Harang R (2019) SeqDroid: obfuscated android malware detection using stacked convolutional and recurrent neural networks. In: Deep learning applications for cyber secu- rity. Springer, pp 197–210
[10] Wang C, Xu Q, Lin X, Liu S (2019) Research on data mining of permissions mode for Android malware detection. Clust Comput 22(6):13337–13350
[11] Pektas¸ A, Acarman T (2019) Learning to detect Android malware via opcode sequences. Neurocomputing 396:599–608
[12] Roopak S, Thomas T, Emmanuel S (2019) Android malware detection mechanism based on Bayesian model averaging. In: Recent findings in intelligent computing techniques. Springer, pp 87–96
[13] Liu P, Wang W, Luo X et al (2021) NSDroid: efficient multi- classification of android malware using neighborhood signature in local function call graphs. Int J Inf Secur 20:59–71
[14] Pektas¸ A, Acarman T (2020) Deep learning for effective Android malware detection using API call graph embeddings. Soft Com- put 24(2):1027–1043
[15] Zou K, Luo X, Liu P, Wang W, Wang H (2019) ByteDroid: android malware detection using deep learning on bytecode sequences. In: Chinese conference on trusted computing and information security. Springer
[16] Taheri R, Ghahramani M, Javidan R, Shojafar M, Pooranian Z, Conti M (2020) Similarity-based android malware detection using Hamming distance of static binary features. Futur Gener Comput Syst 105:230–247
[17] Alzaylaee MK, Yerima SY, Sezer S (2020) DL-Droid: Deep learning based android malware detection using real devices. Comput Secur 89:101663
[18] Bakour K, U¨ nver HM, Ghanem R (2019) The Android malware detection systems between hope and reality. SN Appl Sci 1(9):1120
[19] Yen Y-S, Sun H-M (2019) An android mutation malware detection based on deep learning using visualization of impor- tance from codes. Microelectron Reliab 93:109–114
[20] Hsien-De Huang T, Kao H-Y (2018) R2-d2: color-inspired con- volutional neural network (CNN)-based android malware detec- tions. In: 2018 IEEE international conference on big data (Big Data). IEEE
[21] Bakour K, U¨ nver HM (2020) VisDroid: android malware classification based on local and global image features, bag of visual words and machine learning techniques. Neural Comput Appl. https://doi.org/10.1007/s00521-020-05195-w
[22] U¨ nver HM, Bakour K (2020) Android malware detection based on image-based features and machine learning techniques. SN Appl Sci 2(7):1–15
[23] Zhang H, Ji Y, Huang W, Liu L (2019) Sitcom-star-based clothing retrieval for video advertising: a deep learning frame- work. Neural Comput Appl 31(11):7361–7380. https://doi.org/10. 1007/s00521-018-3579-x
[24] Onwuzurike L, Mariconti E, Andriotis P, Cristofaro ED, Ross G, Stringhini G (2019) MaMaDroid: detecting android malware by building markov chains of behavioural models (extended version). ACM Trans Priv Secur (TOPS) 22(2):14
[25] Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K, Sie- mens C (2014) Drebin: effective and explainable detection of android malware in your pocket. In: Ndss

[26] Zhou Y, Jiang X (2012) Dissecting android malware: character- ization and evolution. In: 2012 IEEE symposium on security and privacy. IEEE
[27] Guardsquare (2020) Optimizing android resources. https://www. guardsquare.com/en/blog/optimizing-android-resources. Acces- sed 26 May 2020
[28] Mallick S (2018) Shape matching using Hu moments. https:// www.learnopencv.com/shape-matching-using-hu-moments-c- python/. Accessed 19 April 2019
[29] Haralick RM, Shanmugam K, Dinstein IH (1973) Textural fea- tures for image classification. IEEE Trans Syst Man Cybern 6:610–621
[30] Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
[31] Bay H, Tuytelaars T, Van Gool L (2006) Surf: speeded up robust features. In: European conference on computer vision. Springer
[32] Alcantarilla PF, Bartoli A, Davison AJ (2012) KAZE features. In: European conference on computer vision. Springer
[33] Rosten E, Drummond T (2006) Machine learning for high-speed corner detection. In: European conference on computer vision. Springer
[34] Calonder M, Lepetit V, Strecha C, Fua P (2010) Brief: binary robust independent elementary features. In: European conference on computer vision. Springer
[35] He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition
[36] Zhang A, Lipton ZC, Li M, Smola AJ (2019) Dive into deep learning. Unpublished Draft. Retrieved 19
[37] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition
[38] Sharma H (2019) ReLU, Leaky ReLU and Softmax basics for neural networks and deep learning. https://medium.com/ @himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and- softmax-basics-for-neural-networks-and-deep-8d9c70eed91e#:*:text=ReLU Accessed 19 Oct 2020
[39] Bakour K, U¨ nver HM, Ghanem R (2019) A deep camouflage: evaluating android's anti-malware systems robustness against hybridization of obfuscation techniques with injection attacks. Arab J Sci Eng 44(11):9333–9347