Scientific African 29 (2025) e02816

Contents lists available at ScienceDirect

Scientific African

journal homepage: www.elsevier.com/locate/sciaf

SNEL-DFF: Android malware detection using Siamese networks with ensemble learning

Atif Raza Zaidi [a], Tahir Abbas [a], Sadaqat Ali Ramay [a], Tariq Shahzad [b], Zahid Hussain Qaisar [c], Muhammad Adnan Khan [d,*], Adnan Abu-Mahfouz [e,f,**], Amin Beheshti [g]

[a] Department of Computer Science, TIMES University, Multan, Pakistan
[b] Department of Computer Engineering, COMSATS University Islamabad, Sahiwal Campus, Sahiwal, 57000, Pakistan
[c] Department of Computer Science, Emerson University, Multan, Pakistan
[d] Department of Software, Faculty of Artificial Intelligence and Software, Gachon University, Seongnam-si 13557, Republic of Korea
[e] NextGen Enterprises and Institutions, Council for Scientific and Industrial Research, Pretoria, South Africa
[f] Department of Electrical Engineering, Tshwane University of Technology, Pretoria 0001, South Africa
[g] Center for Artificial Intelligence, Macquarie University, Sydney NSW 2109, Australia

ARTICLE INFO

Editor name: B. Gyampoh

Keywords:
Mobile cybersecurity
Android based malware
Feature enrichment
Cascade forest
Deep ensemble learning

ABSTRACT

This paper proposes a new model simply known as Siamese Networks of Optimal Ensemble Learning with Deep Forest Feature (SNEL-DFF). The proposed model has the Deep Forest Feature extraction feature because of the complexity that is present in the data and to enhance the proficiency of the detection system. The feature vectors used in this study includes 215 attributes in android applications which are derived from samples sourced from Drebin dataset. Some of the performance evaluation results have been highlighted revealing that the proposed model yielded an accuracy of 0.99. The precision of 0.98 shows its ability to avoid miss-identification of negatives and the recall of 0.99 proves the effectiveness of using it for detection of the real malware samples. The F1 score is 0.99 and ROC-AUC value of 0.99 indicating the model has achieved 99% accuracy which points to the fact that it is balanced and provides a superior performance. These findings vindicate the hypothesis that SNEL-DFF has strong predictive accuracy as compared to the conventional machine learning algorithms. The proposed technique utilizes Siamese networks, deep forest feature enhancement, and ensemble learning, which makes it perform better than its competitors in terms of various evaluation criteria.

Introduction

The proliferation of Android devices has revolutionized the mobile technology landscape, making it the dominant operating system in the smartphone market [1]. However, this widespread adoption has also made Android a prime target for malware developers. According to recent security reports, the number of distinct Android malware samples has escalated sharply, surpassing 6.2 million by the end of 2023. This rapid increase in malware variants poses a significant threat to mobile security, as attackers continuously develop new techniques to bypass existing defense mechanisms [2,3].

* Corresponding author.
** Corresponding author at: NextGen Enterprises and Institutions, Council for Scientific and Industrial Research, Pretoria, South Africa.
E-mail addresses: adnan@gachon.ac.kr (M.A. Khan), aabumahfouz@csir.co.za (A. Abu-Mahfouz).

https://doi.org/10.1016/j.sciaf.2025.e02816
Received 11 September 2024; Received in revised form 11 June 2025; Accepted 16 June 2025
Available online 22 July 2025

Traditional malware detection methods, such as signature-based detection and heuristic analysis, have been the cornerstone of mobile security for years. However, these approaches have significant limitations. Traditional techniques like signature matching and manual analysis have their limitations; they are inaccurate, slow, only effective at recognizing known malware by its specific patterns, and fail at detecting new and unseen malware variants. [4–6]. Signature-based methods are ineffective against zero-day attacks and polymorphic malware, where the malicious code mutates to evade detection. Heuristic analysis, although capable of identifying previously unseen threats, suffers from high false positive rates and is often too slow to keep up with the increasing volume and variety of malware [7–9].

Given the inadequacies of these traditional methods, there is an urgent need for more advanced, adaptive malware detection techniques [10,11]. Recent advancements in machine learning, particularly deep learning, have opened new avenues for malware detection. These techniques excel at identifying complex patterns and behaviors in large datasets, making them well-suited for detecting sophisticated Android malware [12–14].

This paper introduces a novel approach to Android malware detection, the Siamese Networks with Ensemble Learning and Deep Forest Feature Enhancement (SNEL-DFF) model. The proposed model integrates three advanced techniques: Siamese networks for similarity learning, ensemble learning for robust classification, and deep forest feature enhancement for improved feature representation. This hybrid approach is designed to address the limitations of existing methods by providing a more accurate, adaptable, and efficient solution for detecting Android malware.

The main contributions of this study are as follows:

- **Development of a Novel Malware Detection Model:** This study proposes a new malware detection framework, SNEL-DFF, that integrates Siamese Networks, Ensemble Learning, and Deep Forest Feature Enhancement to improve detection accuracy and reduce false positives.
- **Enhanced Feature Representation:** The model leverages deep forest feature extraction to capture complex patterns in the data, leading to a more refined and informative feature set for the classification of Android applications.
- **Robust and Scalable Detection:** By combining the strengths of multiple machine learning techniques, the proposed model achieves a high detection accuracy and can be efficiently scaled to handle large datasets, making it suitable for real-world applications.
- **Comprehensive Performance Evaluation:** The model has been extensively tested on a diverse dataset derived from the Drebin project, demonstrating superior performance metrics compared to traditional machine learning models.
- **Adaptability to New Threats:** The inclusion of Siamese Networks enables the model to learn similarities between malware and benign applications, enhancing its ability to detect new and previously unseen malware variants.

Background and related work

The identification of Android malware has become an important issue for research because Android devices are becoming increasingly popular, and malware is becoming more complex. There are different strategies that have been put forward depending on their type, such as conventional machine learning methods and deep learning. This section provides a brief analysis of the literature in this domain to show how the current study contributes to filling the gaps in knowledge that exist from previous research efforts. Xu et al. [15] have put forward Mobsafe, a cloud computing-based forensic analysis framework for mobile applications using the data mining approach. This approach was designed to take advantage of the computing capabilities of the cloud to perform computations on big data sets of mobile applications, but it had issues regarding data privacy and real-time data analysis. Mylonas et al. [16] studied security awareness in smartphone platforms and the need for teaching users about malware risk. However, it is important to note that user behavior alone cannot be enough to prevent the spread of malware as technical solutions are required for a more effective protection.

Pandita et al. [17] designed Whyper, an automated risk assessment tool for the mobile application. The following tool was useful in determining the privacy risks of apps by evaluating the permissions they required. While permission-based methods are very useful, permission based methods can be easily bypassed by today's smart malware that hardly request for any permission. In another work, Fang et al. [12] discussed the performance of permission-based detection of Android malware, its weaknesses, and possible ways of enhancing the efficiency of the technique. But even such strategies involved a large number of false positives. Shabtai et al. [18] put forward the idea of the detection of mobile malware through the examination of the network traffic anomalies of applications. This behavior-based approach was rather revolutionary but was not without its problems and the major one was that it was difficult to model good traffic in the network. Yerima et al. [19] put forward an ensemble learning technique for high accurate Android malware detection. Their methodology involved the use of multiple classifiers to enhance the detection rate but this came at the cost of consuming a lot of resource.

Yuan et al. [20] proposed Droid-sec, a deep learning methodology for Android malware detection. This approach showed that deep learning was viable in this context but it was still limited by the availability of large labeled training data. Saxe and Berlin [21] proposed a malware detection technique based on deep neural networks by utilizing two-dimensional binary program characteristics. Their method has high detection rates, but their method is computationally costly. Mchaughlin et al. [22] introduced a deep learning Android malware detection model that employed different deep learning approaches. That way, although the system is effective, it is also complicated, and therefore poses a problem of applicability on resource-constrained devices.

Subsequent researches have attempted to expand the knowledge on enhanced methods of Android malware identification. For example, Almarshad et al. [5] employed the Siamese shot learning method for security, which showed better performance in terms of

Table 1
Comparative table of previous studies.

Reference	Technique	Contribution	Limitations	Outcomes
Xu et al. [15]	Data Mining	Cloud-based forensic analysis	Privacy and real-time analysis issues	Leveraged cloud computing for analysis
Pandita et al. [17]	Risk Assessment	Automated risk assessment	Limited by permission-based analysis	Identified potential security risks
Fang et al. [12]	Permission-Based Detection	Evaluated permission-based detection	High false-positive rates	Proposed countermeasures
Shabtai et al. [18]	Behavior Analysis	Detection through network behavior analysis	Modeling benign network behavior	Innovative behavior-based detection
Yuan et al. [20]	Deep Learning	Deep learning framework for detection	Requires large datasets	Demonstrated potential of deep learning
Yerima et al. [19]	Ensemble Learning	High accuracy detection	Computationally intensive	Improved detection accuracy
Saxe & Berlin [21]	Neural Networks	Two-dimensional binary program features	Computationally intensive	High detection rates
Mchaughlin et al. [22]	Deep Learning	Combined deep learning techniques	Complex system	Effective but resource-heavy
Almarshad et al. [5]	Siamese Learning	Siamese shot learning technique	Adaptability issues	Improved detection accuracy
Bayazit et al. [7]	State-of-the-Art Review	Modern learning models review	Highlights need for robust methods	Comprehensive review
Alamro et al. [3]	Ensemble Learning	Optimal ensemble learning	Scalability issues	High accuracy

precision and a reduced number of false positives. However, their approach does not seem to be comprehensive for the adaptability to new malware variants as the previous step proposed. Building upon the work of Bayazit et al. [7], which offered a state-of-art on protecting android devices from malware attacks, including modern learning models and their challenges. This comprehensive review emphasized the need for further development of the more efficacious and elastic detection approaches. Alamro et al. [3] presented an optimal ensemble learning model for conducting automated analysis of Android malicious applications with high accuracy but this model has a problem of scalability. First, there are two theoretical gaps in the literature that are worthy of mention: The first one is that previous studies have failed to provide a conclusive definition of what constitutes ICD; secondly, the literature has not identified any comparative analysis between ICD and traditional CBR. It is often the case that methods that use permissions and behavior analysis contain a large number of false positives. Even though deep learning has been demonstrated to achieve high accuracy, these approaches are not so transferable to new and diverse malware forms. Some of the proposed methods are computationally extensive, and cannot be deployed easily on devices like smart phones. It is therefore necessary to come up with better feature representation methodologies that can make it easier to differentiate between harmless and malicious apps. To fill these gaps, this research proposes the following: Incorporation of the deep learning algorithms such as Siamese Networks, Optimal Ensemble Learning, and Deep Forest Feature Enhancement in creating an efficient and adaptive Android malware detection system (see Table 1).

Methodology

In this section, the approach employed in the development and assessment of our framework for the Android malware identification is discussed. In this way, we incorporate more sophisticate deep learning methodologies such as Siamese Networks, Optimal Ensemble Learning, and Deep Forest Feature Enhancement for developing a stable and effective detection model. The dataset utilized for the purpose of training and testing is the Android Malware Data obtained from Drebin [6].

Dataset

The data set applied in this research is the Android Malware Data, which is acquired from Drebin [6]. This dataset contains a large number of feature vectors from the Android applications collected from the malware and legitimate sources. Mobile malware: malicious software that attacks hand held devices such as mobile phones or wireless enabled PDAs, resulting in the system failure and loss or theft of identity information. When using wireless phones and PDA networks, these devices and their connectivity have become more frequent and elaborate, the safety and security of the phone from electronic attack in the form of viruses or other forms of malware cannot be easily secured. The dataset has feature vectors of 215 attributes from 15,036 applications 5560 of which belongs to the Drebin project and the remaining 9476 are benign applications. As a result, this rich data set can offer the basis for training and evaluating the proposed malware detection model. Table 2 provides a detailed overview of the top 10 features identified by the Deep Forest model as having the highest importance in the dataset. These features, which include permissions, API

Table 2
Top extracted features by Deep Forest (DFF) with relative importance scores.

Feature name	Importance	Data type
SEND_SMS	0.071622	Boolean
android.telephony.SmsManager	0.042968	Boolean
READ_PHONE_STATE	0.042046	Boolean
attachInterface	0.039656	Boolean
android.os.Binder	0.038371	Boolean
onServiceConnected	0.037269	Boolean
transact	0.034955	Boolean
ServiceConnection	0.034862	Boolean
TelephonyManager.getDeviceId	0.028427	Boolean
Ljava.lang.Class.getCanonicalName	0.025086	Boolean
bindService	0.024615	Boolean
INTERNET	0.022029	Integer
Ljava.lang.Class.cast	0.021466	Boolean
RECEIVE_SMS	0.020123	Boolean
READ_SMS	0.019121	Boolean

calls, intents, activities, services, and other critical attributes, are described alongside their respective data types and importance scores.

Additional feature visualizations including correlation matrices, histograms, KDE plots, density overlays, and pairwise comparisons are presented in the Supplementary Material (Figures S1–S5).

Proposed model: Siamese networks of optimal ensemble learning with deep forest feature (SNEL-DFF)

The proposed model that is SNEL-DFF involves the Siamese Networks, Optimal Ensemble Learning and Deep Forest Feature Enhancement for detection of Android malware. The model was designed to work as a whole where each component contributes its core functionality to provide an effective solution to the malware detection problem.

Deep Forest Feature Enhancement

Deep Forest is a pre-processing point-based learning method generally based on decision trees that improve feature representation through stacked consecutively random forests. In the cascade, each layer takes raw features and outputs as input from the previous layer of the cascade. Let $h^{(l)}$ be the output of the lth layer; we define $h^{(0)} = x$. The output of the lth layer is given by:

$$h^{(l)} = f^{(l)}(x, h^{(l-1)}) \dots \tag{1}$$

where $f^{(l)}$ is the function representing the lth layer. The final representation $h^{(L)}$ is used for classification.

Deep Forest uses layered random forests to detect complex data patterns, enhancing features for subsequent stage. Deep Forest is able to learn non-linear relationships and complex structures within the dataset. In the case of malware detection, this capability is particularly useful as it allows the model to distinguish small variations in the behavior of benign and malicious applications, where other methods may fail. Additionally, the cascading structure of Deep Forest enables the combination of different representations of the input data, enhancing the extracted features and improving the effectiveness of the classification task (see Fig. 1).

Siamese networks

Siamese Networks refer to a category of neural networks that consist of two identical networks with parallel configurations to analyze two different inputs (see Fig. 2). The aim is to predict a single value for a given pair of inputs, so we need to acquire the function of similarity measures between two inputs. If two input vectors are given as $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively, then the network outputs two vectors $\mathbf{h}_1$ and $\mathbf{h}_2$, which represent the embeddings of the input vectors.

The network is trained to minimize the contrastive loss, defined as:

The loss function is defined as:

$$L(x_1, x_2, y) = (1-y)\frac{1}{2}D(h_1, h_2)^2 + y\frac{1}{2}\max(0, m - D(h_1, h_2))^2 \dots \tag{2}$$

where y is a binary label indicating whether x_1 and x_2 are similar, $D(h_1, h_2)$ is the Euclidean distance between h_1 and h_2, and m is a margin parameter.

Optimal ensemble learning

For our dataset, we utilize a Deep Forest model for feature extraction, followed by a Siamese Network to measure similarity between transformed features. The final classification is performed using an ensemble approach with a logistic regression meta-learner, leveraging the strengths of both base models. The combination of Siamese Networks with Ensemble Learning techniques presents a new way of detecting malware. Siamese Networks are designed to look for similarities between two sequences and, as such, are very effective at identifying small shifts in behavior between normal and malicious applications. On their own, they may not capture all the complex interactions in the malware feature spaces. What makes this approach powerful is the integration

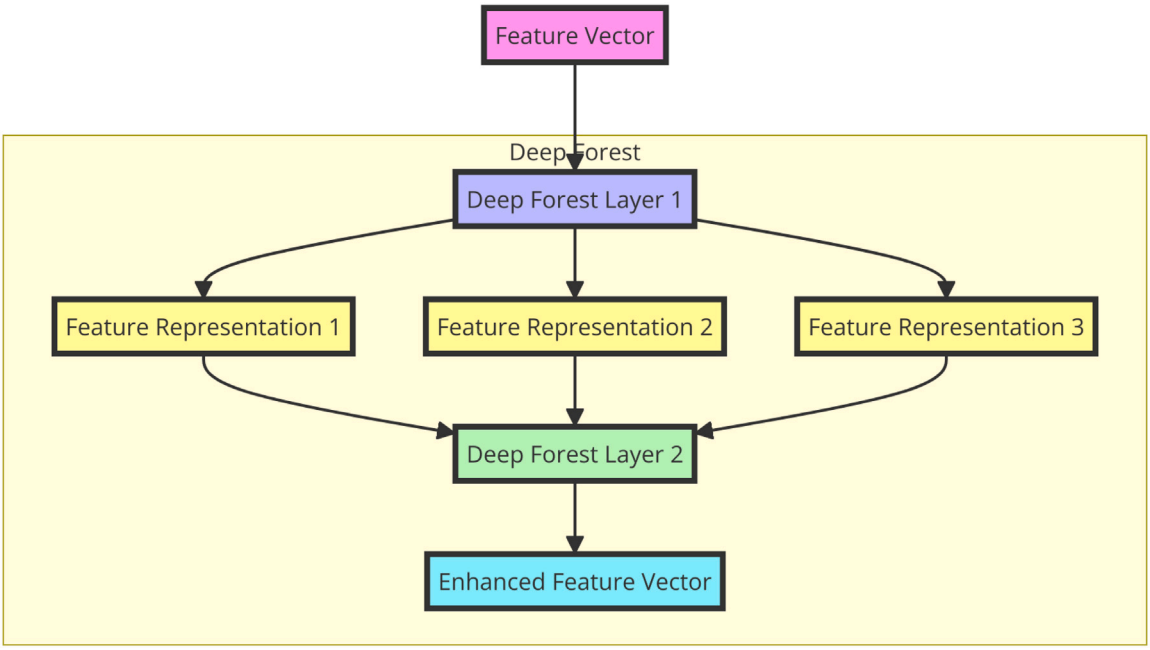

Fig. 1. Internal Architecture of Deep Forest Feature Extraction.

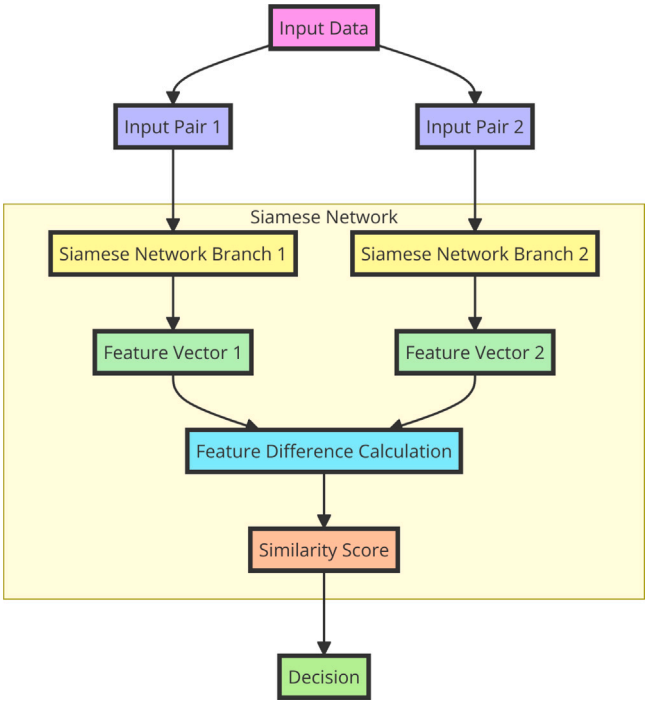

Fig. 2. Internal Architecture of the Siamese Network.

of Ensemble Learning, specifically Deep Forest Feature Enhancement, which provides for robust feature extraction and diverse decision-making.

In ensemble learning, several classifiers are created to optimize the classification of the sample data. In the proposed model, we make improvements to a classifier set, which is an optimal combination of an array of base classifiers utilizing a meta-learner (see

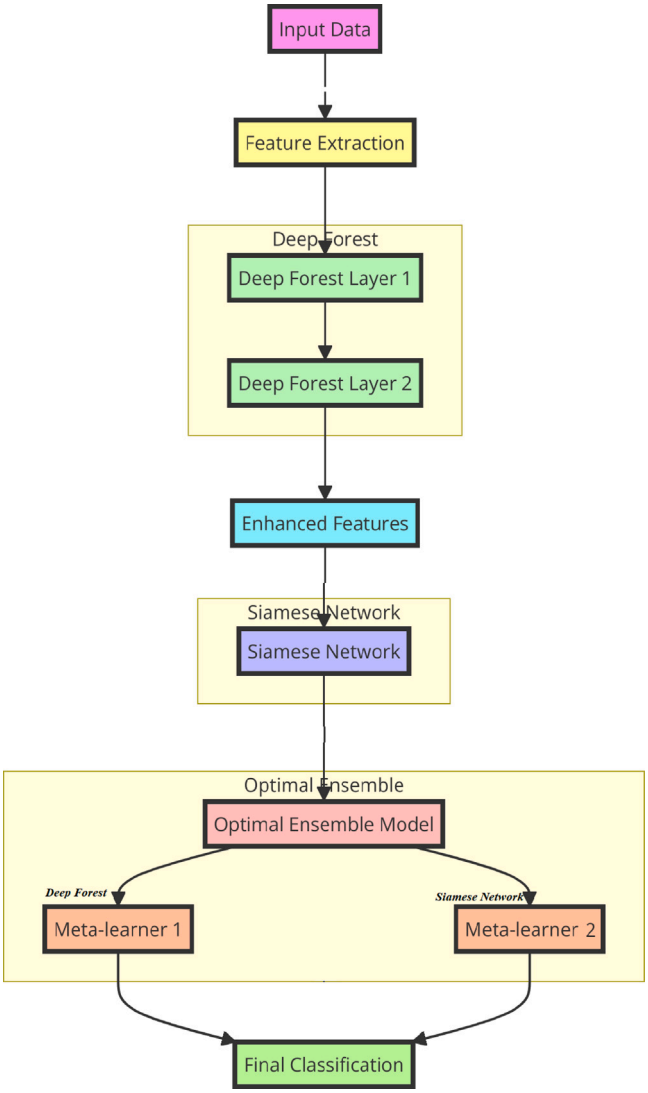

Fig. 3. Overall architecture.

Fig. 3). Let $\{f_1, f_2, \ldots, f_n\}$ be the set of base classifiers, and let $\mathbf{x}$ be an input vector. The ensemble output is given by:

$$F(\mathbf{x}) = g(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})) \ldots \tag{3}$$

where g is the meta-learner that combines the outputs of the base classifiers. The meta-learner can be a simple average, weighted average, or a more complex model like a neural network.

Performance metrics

There are several metrics that are used to measure the effectiveness of the model in machine learning. In the context of this research, different and widespread performance indices are employed to evaluate the proposed SNEL-DFF model. Some of the internal metrics include accuracy, precision, recall, F1 score, and ROC AUC.

Accuracy

Accuracy measures the degree of correct predictions to the total cases in the dataset. The formula for accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Algorithm 1: Proposed Model: SNEL-DFF

Input: Dataset D with feature vectors x_i and labels y_i
Output: Final classification results
Initialization *Initialize Deep Forest parameters Θ, Siamese Network parameters θ_1 and θ_2, Ensemble model parameters Ω;*
Step 1: Deep Forest Training
  Train CascadeForestClassifier on training data;
  Save model and use predictions as transformed features;
end
Step 2: Feature Extraction
  for *each instance x_i in D* **do**
    Compute feature vector $h_i = f(x_i; \theta_1)$;
    Store feature vector h_i for Siamese Network input;
  end
end
Step 3: Siamese Network Training
  Create and train a Siamese Network using transformed features;
  Sub-step
    Define base network with Dense layers;
    Generate pairs for training, train for 10 epochs;
  end
  for *each pair (x_i, x_j) in the training set* **do**
    Compute feature vectors $h_1 = f(x_i; \theta_1)$ and $h_2 = f(x_j; \theta_2)$;
    Compute Euclidean distance $D(h_1, h_2) = \|h_1 - h_2\|_2$;
    Compute contrastive loss:

$$L_{\text{contrastive}} = (1 - y_{ij})\frac{1}{2}D(h_1, h_2)^2 + y_{ij}\frac{1}{2}\max(0, m - D(h_1, h_2))^2$$

    Update Siamese Network parameters θ_1 and θ_2 using backpropagation;
  end
end
Step 4: Stacking Ensemble
  Use transformed features for meta-learning;
  Create DataFrame with predictions from Deep Forest and Siamese Network;
  Train Logistic Regression as meta-learner on these predictions;
  Update ensemble model parameters Ω using gradient descent;
end
Step 5: Final Classification
  for *each instance x_i in the test set* **do**
    Compute feature vector $h_i = f(x_i; \theta_1)$;
    Compute enhanced feature vector $h_i^{(L)} = f^{(L)}(h_i; \Theta)$;
    Compute classification result $y_i = g(f_1(h_i^{(L)}), \ldots, f_K(h_i^{(L)}); \Omega)$;
  end
end

Precision

Precision is the number of correctly predicted positive observations over the total predicted positive observations:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

Recall (True positive rate)

Recall is the percentage of actual positive results that are correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

F1-score

The F1-score is the harmonic mean of precision and recall:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

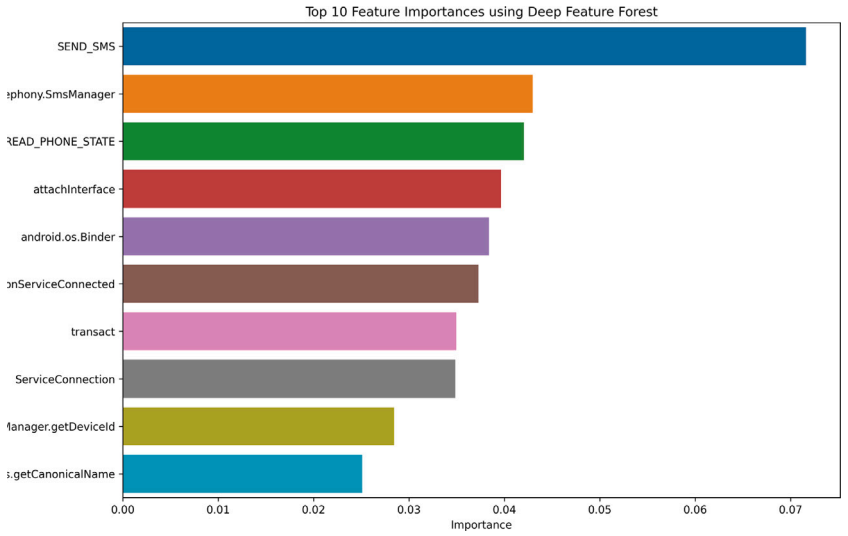

Fig. 4. Features Extracted by DFF.

ROC-AUC

The ROC curve plots the true positive rate against the false positive rate. The AUC represents the area under this curve:

$$\text{AUC} = \int_0^1 TPR(f)\, d(FPR(f)) \tag{8}$$

where TPR is True Positive Rate and FPR is False Positive Rate.

Specificity

Specificity measures how well the model identifies negatives:

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{9}$$

Confusion matrix

A confusion matrix is a 2×2 table that reports performance results of a classification model:

- **True Positives (TP)**: Actual positive observations correctly predicted.
- **True Negatives (TN)**: Actual negative observations correctly predicted.
- **False Positives (FP)**: Negative observations incorrectly predicted as positive.
- **False Negatives (FN)**: Positive observations incorrectly predicted as negative.

The confusion matrix highlights correct and incorrect predictions for each class.

Results and discussions

Here we are showing various performance indexes to evaluate the efficiency of the proposed model. The value metrics of a model that is in these categories are Accuracy, Precision, Recall, F1-Score, and ROC-AUC, each of which provides a different view of the model. By these measures and visualizations, it is our aim to demonstrate that the introduced model can indeed help in identifying Android malware and that the model is dependable and performs well when it comes to its deployment.

DFF feature extraction

In this subsection, we explicate how the DFF component of the proposed model detects and extracts the features. DFF technique is particularly useful when it comes to extracting credible and high level features from the input data and this has a way of boosting the model's capacity in accurately identifying malware (see Fig. 4).

Additional details on the top-ranked features, including their importance scores and functional roles, are provided in Supplementary Material (Table 1).

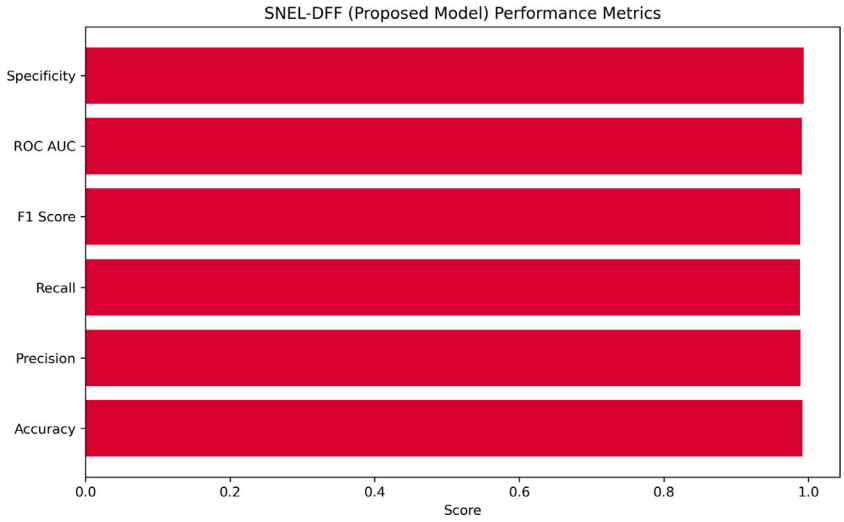

Fig. 5. Metrics of Proposed Model: SNEL-DFF.

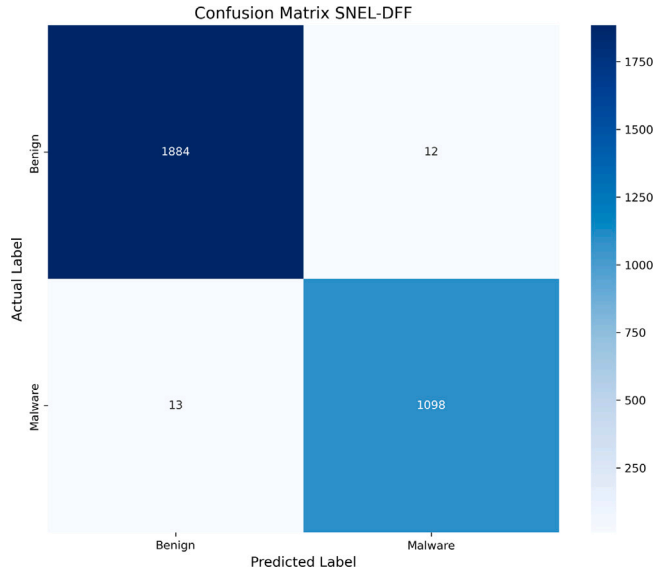

Fig. 6. Confusion Matrix of Proposed Model.

Performance of proposed model

The performance measurement is conducted by using accuracy, precision, recall, F1-score, and ROC-AUC as indicators. The findings show how accurate our model is in identifying the infected Android applications. The precise numerical results are given to present the availability of quantitative data to explain the model's effectiveness (see Fig. 5 and Table 3).

The confusion matrix, illustrated in Figure below, helps create a clear understanding of the true positive, true negative, false positive, and false negative rates of the proposed model. It reveals that the model proposed in this paper is able to maintain a good balance in identifying benign and malicious applications with 1098 true positives and 1884 true negatives. The rate of false positives (12) is low and consists of benign applications which are incorrectly labeled as malware, which in real life could result in false alarms or disruptions. The rate of false negatives (13), i.e., malware that has been incorrectly identified as benign, is a more serious problem because of the possibility of the presence of malicious activity going undetected (see Fig. 6).

Comparative analysis

The evaluation of the proposed SNEL-DFF model required a thorough comparison with widely used machine learning algorithms including Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (DT), and K-Nearest Neighbors (KNN). The results

Table 3
Performance metrics of the proposed SNEL-DFF model.

Metric	Value
Accuracy	0.992686
Precision	0.989189
Recall	0.988299
F1-score	0.988744
ROC–AUC	0.990985
Specificity	0.993671

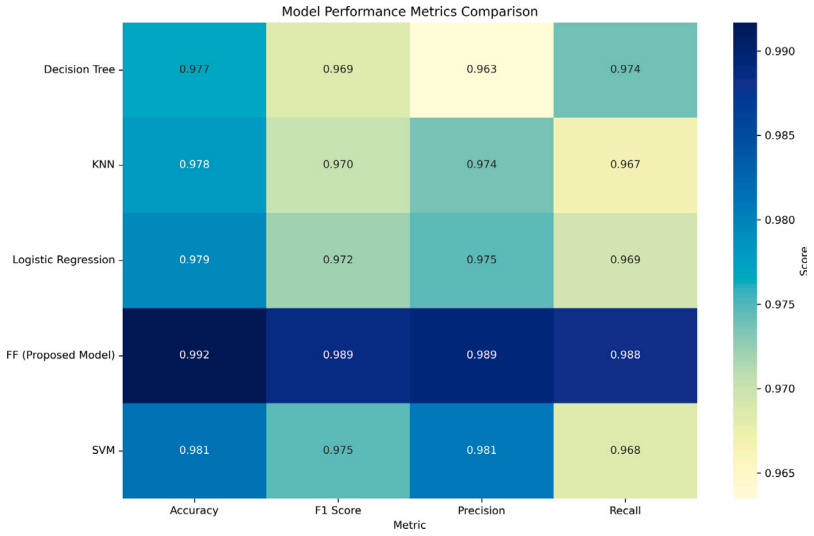

Fig. 7. Overall Performance Comparison with Machine Learning Algorithms.

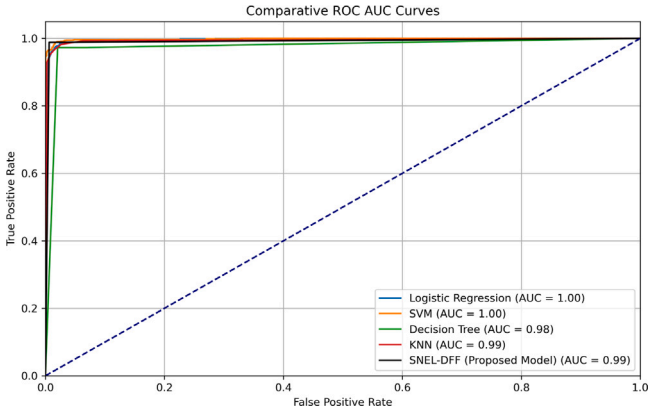

Fig. 8. ROC-AUC Comparison of Different Models.

show that SNEL-DFF outperforms all baseline models in all key performance metrics including accuracy, precision, recall, F1-score, and ROC-AUC (see Figs. 7 and 8).

Additional comparative plots and numerical tables are provided in the Supplementary Material (Figures S6–S11).

Extended discussion/interpretation

Here we invoke details about the implemented and proposed SNEL-DFF model that comprises a combination of Siamese Networks and Optimal Ensemble Learning with Deep Forest Feature and compare its runtime analysis and detection score in the context of Android malware. As it has been demonstrated while representing various coefficients in the previous section, the applicability of the proposed approach is higher than the applicability of traditional machine learning models in several aspect. The SNEL-DFF

model achieved an accuracy of 99% and this is considered to be an excellent result. Precision of 98%, recall of 99%, For F1 score 99%(avg) and specificity was 99%. The ROC-AUC of 99% indicates that SNEL-DFF is in very good balance in distinguishing between benign and malicious applications. One of the major challenges in malware detection datasets is class imbalance when number of benign samples is usually greater than malicious samples. The SNEL-DFF model solves this problem by using class-weighted loss functions during training to guarantee that the minority class (malicious samples) is adequately sampled. In addition, the feature enhancement by means of Deep Forest enables the discovery of small shifts in the minority class, which improves the detection. All these mechanisms simultaneously play the role in enhancing the model robustness and balance of the model's performance according to the high ROC-AUC value. Such statistics demonstrate that the model works great in terms of identifying malware, and in terms of not tagging well-behaving applications, as malicious and, at the same time, does not overlook many applications, which might be potentially malicious. The high value of recall means that the model includes the greatest number of actual malware instances while the high value of precision means that it has a low false positive ratio. Last but not least, the extraction component called Deep Forest Feature (DFF) also contributes to improving the model's performance. This is made possible by applying decision tree-based feature extraction at multiple layers in the DFF design, where complex patterns and interactions in the data are captured that might not be discernable by other feature extraction techniques In essence, DFF provides a much more detailed and informative set of features for the subsequent classification models to work with, which greatly enhances the predictive accuracy of the models. Siamese network architecture is able to learn similarity metrics and therefore well suited for the task of dealing with variability of the data that is typical of malware. Recent approaches using dual Siamese networks on image-based malware representations further validate this capability [23]. Paired with the fact that the identification of the data is based on the comparison of pairs of applications, the model is capable of improving its generalization and, therefore, its resistance to new types of malware. The approach of ensemble learning, as applied in our model, means that several classifiers are involved and all of them are incorporated into the final model because it is known that each classifier has its strength and integrating all of them would make the model stronger [24]. It not only enhances the average performance but also offers protection from overfitting, which is crucial when working with large and multidimensional data sets, such as utilized in malware classification.

When we compared the performance of the proposed model, SNEL-DFF, with the other models such as LOGISTIC REGRESSION, SVM, DECISION TREE, and KNN, our model has a much higher classification accuracy. These improvements are primarily as a result of an effective feature extraction method of SNEL-DFF and the insertion of diverse learning algorithms. It preserves the strength of individual base classifiers and makes sure that the model delivers a dependable detection mechanism through incorporating the relevant components of the base classifiers in the ensemble. One of the characteristics of machine learning is that it learns on other data that have not been used before and is able to predict well: this is why it is so good at protecting users from new, constantly emerging viruses and other types of malware, and why it offers constant protection.

Conclusion

In conclusion, the designed SNEL-DFF model is proposed for efficient and accurate identification of Android malware samples. The model suggested here proves to be capable to bear the benefits of Siamese networks, deep forest feature extraction as well as the optimal ensemble learning to do incredibly well in the different several suggested evaluation metrics. This implies that its ability to detect the malware effectively and efficiently should enable it to be a useful weapon in the fight against malware.

With this we must affirm that the proposed SNEL-DFF model has been shown to give a decent performance but there is probable always a way to improve on this. Other avenues for future research could also specify the possible utilization of other techniques of feature extraction to the enhancement of the function of the model. Moreover, we could have better understood the model's versatility and robustness if it had been trained and tested on a wider variety of malware and deployment settings .

CRediT authorship contribution statement

Atif Raza Zaidi: Proposed the model, Writing – original draft, Drafted pictures and tables. **Tahir Abbas:** Proposed the model, Formal analysis, simulation, Writing – original draft, Revisions, Improved the quality of draft. **Sadaqat Ali Ramay:** Proposed the model, Writing – original draft, Revisions, Improved the quality of draft. **Tariq Shahzad:** Formal analysis, simulation, Writing – original draft, Drafted pictures and tables. **Zahid Hussain Qaisar:** Proposed the model, Writing – review & editing. **Muhammad Adnan Khan:** Formal analysis, simulation,Supervision, Drafted pictures and tables,Revisions, Improved the quality of draft. **Adnan Abu-Mahfouz:** Formal analysis, simulation, Writing – review & editing, Supervision, Revisions, Improved the quality of draft. **Amin Beheshti:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

All authors have read and agreed to the published version of the manuscript.

Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.sciaf.2025.e02816.

References

[1] P.V. Agrawal, D.D. Kshirsagar, Information gain-based feature selection method in malware detection for MalDroid2020, in: 2022 International Conference on Smart Technologies and Systems for Next Generation Computing, ICSTSN, IEEE, 2022.

[2] Mohammed Nasser Al-Andoli, Kok Swee Sim, Shing Chiang Tan, Pey Yun Goh, Chee Peng Lim, An ensemble-based parallel deep learning classifier with PSO-bp optimization for malware detection, IEEE Access 11 (2023).

[3] Hayam Alamro, Wafa Mtouaa, Sumayh Aljameel, Ahmed S. Salama, Manar Ahmed Hamza, Aladdin Yahya Othman, Automated android malware detection using optimal ensemble learning approach for cybersecurity, IEEE Access 11 (2023).

[4] Ghadah Aldehim, Munya A. Arasi, Majdi Khalid, Sumayh S. Aljameel, Radwa Marzouk, Heba Mohsen, Ishfaq Yaseen, Sara Saadeldeen Ibrahim, Gauss-mapping black widow optimization with deep extreme learning machine for android malware classification model, IEEE Access 11 (2023).

[5] Fahdah A. Almarshad, Mohammed Zakariah, Ghada Abdalaziz Gashgari, Eman Abdullah Aldakheel, Abdullah I.A. Alzahrani, Detection of android malware using machine learning and siamese shot learning technique for security, IEEE Access 11 (2023).

[6] Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, Konrad Rieck, Drebin: Efficient and explainable detection of android malware in your pocket, in: 21th Annual Network and Distributed System Security Symposium, NDSS, 2014.

[7] Esra Calik Bayazit, Ozgur Koray Sahingoz, Buket Dogan, Protecting android devices from malware attacks: A state-of-the-art report of concepts, modern learning models and challenges, IEEE Access 11 (2023).

[8] G. Canfora, F. Mercaldo, C.A. Visaggio, Mobile malware detection using op-code frequency histograms, in: 2015 12th International Joint Conference on E-Business and Telecommunications, ICETE, IEEE, 2015.

[9] Leonardo da Costa, Vitor Moia, A lightweight and multi-stage approach for android malware detection using non-invasive machine learning techniques, IEEE Access 11 (2023).

[10] O.E. David, N.S. Netanyahu, Deepsign: Deep learning for automatic malware signature generation and classification, in: Proc. of the International Joint Conference on Neural Networks, IJCNN, 2015, pp. 1–8.

[11] DroidDetector: A deep learning based android malware detection engine, 2015, http://analysis.droid-sec.com.

[12] Z. Fang, W. Han, Y. Li, Permission based android security: Issues and countermeasures, Comput. Secur. 43 (2014) 205–218.

[13] Ch.-Y. Huang, Y.-T. Tsai, C.-H. Hsu, Performance evaluation on permission-based detection for android malware, Adv. Intell. Syst. Appl. 2 (2013) 111–120.

[14] Lu Huang, Jingfeng Xue, Yong Wang, Dacheng Qu, Junbao Chen, Nan Zhang, Li Zhang, EAODroid: Android malware detection based on enhanced API order, Chin. J. Electron. 32 (5) (2023).

[15] J. Xu, Y.-T. Yu, Z. Chen, B. Cao, W. Dong, Y. Guo, J. Cao, Mobsafe: Cloud computing based forensic analysis for massive mobile applications using data mining, Tsinghua Sci. Technol. 18 (4) (2013) 418–427.

[16] A. Mylonas, A. Kastania, D. Gritzalis, Delegate the smartphone user? Security awareness in smartphone platforms, Comput. Secur. 34 (2013) 47–66.

[17] R. Pandita, X. Xiao, W. Yang, W. Enck, T. Xie, Whyper: Towards automating risk assessment of mobile applications, in: Proceedings of the 22nd USENIX Security Symposium (USENIX Security), 2013, pp. 527–542.

[18] A. Shabtai, et al., Mobile malware detection through analysis of deviations in application network behavior, Comput. Secur. 43 (2014) 1–18.

[19] S.Y. Yerima, S. Sezer, I. Muttik, High accuracy android malware detection using ensemble learning, IET Inf. Secur. 9 (6) (2015) 313–320.

[20] Z. Yuan, et al., Droid-sec: deep learning in android malware detection, in: Proceedings of the 2014 ACM Conference on SIGCOMM, 2014.

[21] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, in: Proc. of the 10th International Conference on Malicious and Unwanted Software, MALWARE, 2015, pp. 11–20.

[22] N. Mchaughlin, J. Martinez del Rincon, B-J. Kang, S. Yerima, Y. Safaei, E. Trickel, Z. Zhao, A. Doupe, G. Joon Ahn, Deep android malware detection, in: Proc of the ACM on Conference on Data and Application Security and Privacy, CODASPY, 2017, pp. 301–308.

[23] Ramu Kuchipudi, Misbah Uddin, T Satyanarayana Murthy, Teja Kiran Mirrudoddi, Mustafa Ahmed, et al., Android malware detection using ensemble learning, in: 2023 International Conference on Sustainable Computing and Smart Systems, ICSCSS, IEEE, 2023, pp. 297–302.

[24] ByeongYeol An, JeaHyuk Yang, Seoyeon Kim, Taeguen Kim, Malware detection using dual siamese network model., CMES Comput. Model. Eng. Sci. 141 (1) (2024).