BILINGUAL PUBLISHING GROUP

Artificial Intelligence Advances

https://journals.bilpubgroup.com/index.php/aia

RESEARCH ARTICLE

Domain Adaptation-Based Deep Learning Framework for Android Malware Detection Across Diverse Distributions

***Shuguang Xiong*[1,2], *Xiaoyang Chen*[3], *Huitao Zhang*[4*], *Meng Wang*[5]**

[1] *Microsoft Inc., Beijing 100102, China*

[2] *Baidu Inc. Beijing 100085, China*

[3] *The Ohio State University, Columbus, OH 43210, United States*

[4] *Northen Arizona University, San Francisco St, AZ 86011, United States*

[5] *Newmark Group, United States*

ABSTRACT

This study addresses the challenge of Android malware detection, a critical issue due to the pervasive threats affecting mobile devices. As Android malware evolves, conventional detection methods struggle with novel or polymorphic malware that bypasses traditional defenses. This research leverages machine learning (ML) and deep learning (DL) techniques to overcome these limitations by adopting domain adaptation strategies that enhance model generalization across different distributions. The approach involves dividing a dataset into distinct distributions and applying domain adaptation techniques to ensure robustness and accuracy despite distribution shifts. Preliminary results demonstrate that domain adaptation significantly improves detection accuracy in target domains not represented in the training data. This paper showcases a domain adaptation-based method for Android malware detection, illustrating its potential to enhance security measures in dynamic environments. The findings suggest that integrating advanced ML and DL strategies with domain adaptation can substantially improve the efficacy of malware detection systems.

Keywords: Component; Android malware detection; Deep learning; Domain adaptation

1. Introduction

Android malware, a malicious software specifically designed to exploit the Android operating system, has become a significant concern in the mobile eco-

*CORRESPONDING AUTHOR:
Huitao Zhang, Northen Arizona University, San Francisco St, AZ 86011, United States; Email: hz345@nau.edu

ARTICLE INFO
Received: 25 March 2024 | Accepted: 26 April 2024 | Published Online: 29 June 2024
DOI: https://doi.org/10.30564/aia.v6i1.6718

CITATION
Xiong, S., Chen, X., Zhang, H., 2024. Domain Adaptation-Based Deep Learning Framework for Android Malware Detection Across Diverse Distributions. Artificial Intelligence Advances. 6(1): 13–24. DOI: https://doi.org/10.30564/aia.v6i1.6718

system [1,2]. With the exponential growth of Android devices globally, the number of malwares targeting these devices has also surged, posing serious threats to users' privacy, financial security, and the overall integrity of the mobile environment. Malware can take various forms, including trojans, ransomware, adware, and spyware, each with distinct mechanisms and detrimental effects. Detecting and mitigating these threats is paramount to ensuring the safety and reliability of mobile platforms, thereby safeguarding user data and maintaining trust in digital services. his issue has become even more pronounced in the post-pandemic era, where the use of electronic devices among younger students has increased significantly due to the widespread adoption of educational technology [3,4]. The importance of Android malware detection cannot be overstated. As mobile devices become increasingly integral to daily life, encompassing functions such as banking, communication, and personal management, the impact of a successful malware attack can be devastating. Moreover, malware often evolves to bypass traditional security measures, necessitating advanced and adaptive detection techniques. Effective malware detection not only protects individual users but also preserves the broader digital infrastructure, preventing large-scale breaches and cyber-attacks.

Traditional methods of malware detection primarily rely on signature-based approaches, where known patterns of malicious code are identified and flagged. While effective against previously encountered threats [5], these methods fall short in detecting novel or polymorphic malware, which can modify its code to evade detection. Heuristic-based techniques, which analyze the behavior of applications to identify suspicious activities, offer an improvement but still face limitations in adaptability and accuracy. The dynamic nature of malware, coupled with the vast and varied landscape of Android applications, calls for more sophisticated detection mechanisms.

In recent years, the advent of machine learning (ML) and deep learning (DL) has revolutionized in many fields [6–8] . For instance, Liu et al. introduces two machine learning strategies designed to simultaneously optimize accuracy and output frequency. The first strategy employs a hybrid model combining a convolutional neural network (CNN), a long short-term memory (LSTM) module, and a regression module. The second strategy utilizes a dual random forest model configuration. Both approaches were applied to data obtained from ultra-wideband (UWB) sensors installed on a highway bridge. The results demonstrate that these strategies surpass current leading methods in both measurement accuracy and output frequency [9]. Qiu et al. conducted a thorough analysis of various clustering algorithms to determine the most effective method for accurately categorizing credit card customers. The accuracy of the clustering models was assessed using the Davies-Bouldin Index, Silhouette Score, and Calinski-Harabasz Index [10]. By leveraging large datasets and advanced algorithms, these techniques can learn intricate patterns and make predictions with high accuracy [11,12]. Semantic wireframe detection aids Android malware detection by parsing code structure and behavior patterns, enhancing feature extraction accuracy and detection model efficiency [13]. Also, machine learning models, such as decision trees and support vector machines, have demonstrated notable success in classifying benign and malicious applications. Deep learning models, particularly neural networks, have further enhanced detection capabilities by automatically extracting and learning features from raw data [14–16], thereby improving the detection of complex and evolving malware. Despite the significant advancements, current ML and DL approaches for Android malware detection often assume that the training and testing data are drawn from the same distribution. This assumption, however, does not hold in real-world scenarios, where the distribution of malware and benign applications can change over time and across different environments. Consequently, models trained on a particular dataset may experience a decline in performance when applied to data with a different distribution. This distribution shift can lead to increased false positives and false negatives, undermining the effectiveness of the detection system and potentially exposing users to

undetected threats.

To address this challenge, the concept of domain adaptation has emerged as a promising solution [17–19]. Domain adaptation techniques aim to improve the generalization of models by adapting them to new, unseen distributions, thereby enhancing their robustness and accuracy. By aligning the feature space of the source domain (training data) with that of the target domain (testing data), these techniques can mitigate the performance degradation caused by distribution shifts. In this study, we propose a domain adaptation-based approach to Android malware detection shown in **Figure 1**. By dividing our dataset into different distributions and applying domain adaptation techniques, we aim to train models on the source domain and evaluate their performance on the target domain. This approach seeks to demonstrate the effectiveness of domain adaptation in maintaining high detection accuracy across varying distributions, ultimately contributing to more resilient and reliable malware detection systems.

This paper is organized as follows: Section 2 reviews the related works on Android malware detection. Section 3 outlines the workflow of the proposed method in this study. Section 4 presents the experimental results and provides a discussion of the findings. Finally, Section 5 offers a comprehensive conclusion of the paper.

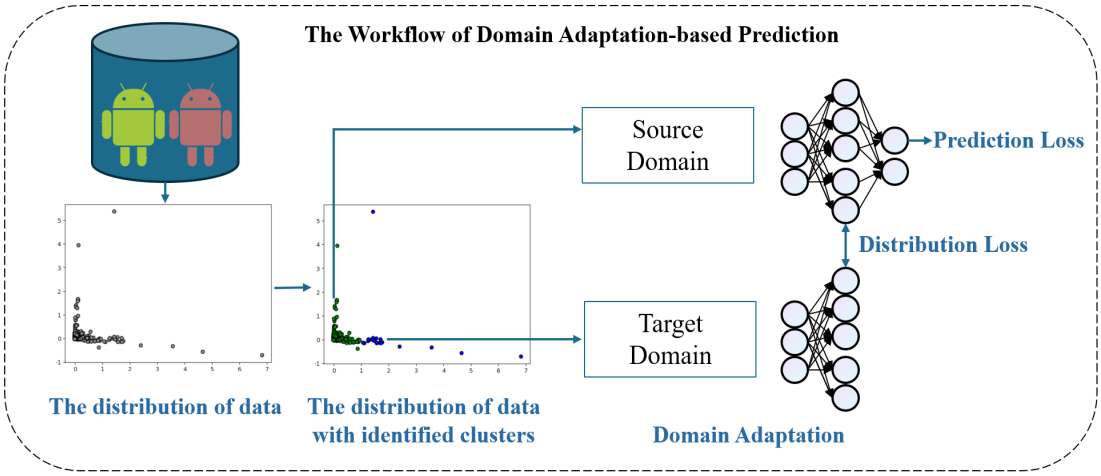

Figure 1. The workflow of the proposed method.

2. Literature review

Android malware detection

Recent research on Android malware detection using machine learning has focused on improving accuracy and efficiency through various techniques due to their excellent performance in many domains [20–22]. For example, Lee et al. applied a genetic algorithm for feature selection, which proved to be more effective than traditional information gain-based methods [23], enhancing the detection capabilities of their machine learning models on a dataset that included 5000 benign apps and 2500 malware instances. Catarina Palma et al. focused on making machine learning models more interpretable in the context of malware detection. They employed techniques like feature selection to pinpoint which app characteristics are most indicative of malware. This approach not only improved the accuracy of malware detection but also made the reasons behind the model's predictions clearer, which is crucial for trust and further improvement of these models [24]. Janaka Senanayake et al. conducted a systematic review of machine learning-based methods for Android malware detection. Their review highlighted the effectiveness of these methods and identified potential areas for improvement, emphasizing the need for ongoing research to address emerging malware threats [25]. Additionally, the architectures and implementations

of distributed file systems can provide efficient data management and storage solutions for Android malware detection, thereby enhancing the performance and reliability of detection systems [26].

Research has shown that advanced photodetection technology and frequency-tunable structures exhibit excellent performance in signal detection under complex conditions [27,28]. This suggests that employing similar domain adaptation techniques could enhance the robustness and adaptability of Android malware detection models across different data distributions, addressing the challenge of model performance variation due to the dynamic nature and continual evolution of malware. While previous studies have demonstrated the effectiveness of machine learning in detecting malware within similar distributions, they often do not address the critical issue of performance across varying data distributions. This paper aims to explore the performance of models trained on one data distribution when applied to another, seeking to improve this aspect through domain adaptation techniques. This approach is expected to lead to more robust and adaptable malware detection systems.

3. Method

3.1 Dataset preparation

The dataset used in this study collected from Kaggle [29] consists of 7,845 entries, each described by 14 features. These features include the name, which identifies the Android application, and tcp_packets, representing the number of TCP packets sent and received. Additionally, dist_port_tcp indicates the number of distinct TCP ports, while external_ips counts the unique external IP addresses contacted. Volume_bytes measures the total volume of data sent and received in bytes, and udp_packets indicates the number of UDP packets. The tcp_urg_packet feature counts TCP packets with URG flags. Source_app_packets and remote_app_packets represent the total packets sent by the application and received from the remote application, respectively. Similarly, source_app_bytes and remote_app_bytes measure the total bytes sent by the application and received from the remote application. There is also a column named source_app_packets.1, which appears to be a duplicate of source_app_packets. DNS_query_times records the number of DNS queries made by the application, and finally, the type feature indicates whether the traffic was benign or malicious.

In the preprocessing phase of our study, we transformed the labels in the dataset into numerical format for compatibility with machine learning models. Specifically, we converted the type feature, which originally categorized traffic as either ‘benign’ or ‘malicious’, into binary labels: ‘0’ for benign and ‘1’ for malicious. Additionally, we applied min-max normalization to the features to ensure that all values fall within the same range, thereby improving the performance and convergence of our machine learning models.

In this study, to establish a dataset with differing distributions of Android malware, we employed the k-means clustering algorithm. K-means [30,31] is a widely used method in data mining that partitions data into k distinct clusters based on feature similarity. Each cluster is defined by the mean of the data points assigned to the cluster, minimizing the variance within each cluster. To determine the optimal number of clusters (k), we utilized Silhouette scores, which measure the quality of the clustering. The Silhouette score is a metric that assesses how similar an object is to its own cluster compared to other clusters. A higher Silhouette score indicates a better-defined clustering. Our analysis determined that the best clustering occurred at $k = 2$. We designated the larger cluster, depicted by green points, as the source domain, and the smaller cluster, represented by blue points, as the target domain. **Figures 2**, **Figures 3**, and **Figures 4** in our study illustrate the PCA distribution of the original data, the curve of Silhouette scores, and the PCA distribution after clustering, respectively.

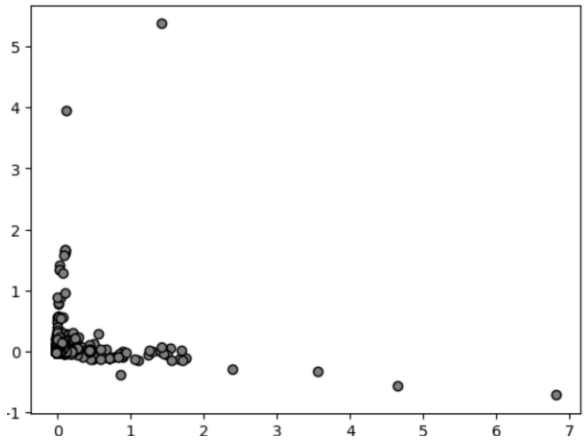

Figure 2. The PCA distribution of the original data.

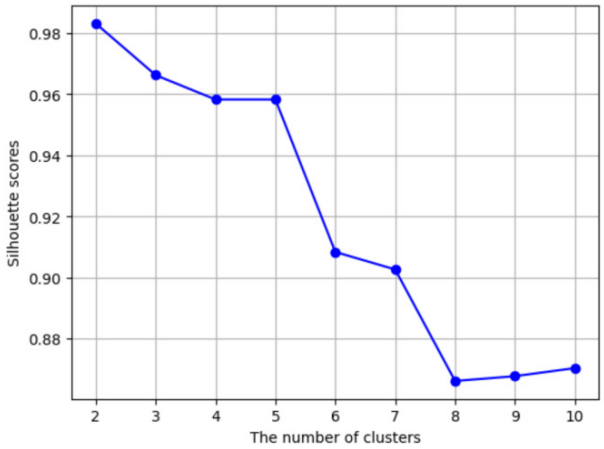

Figure 3. The curve of Silhouette scores.

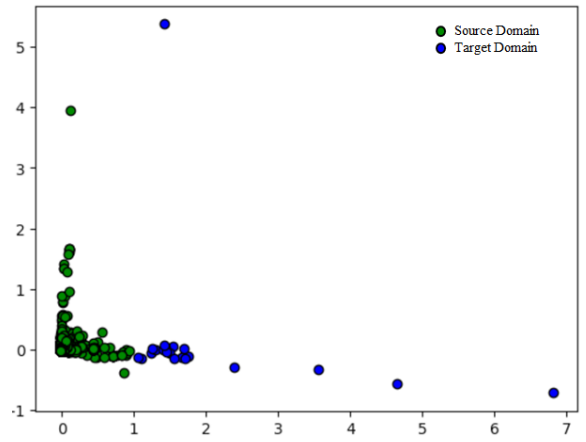

Figure 4. The PCA distribution of the data with identified clusters.

3.2 The introduction of domain adaptation-based ANN model

Artificial neural networks

Artificial Neural Networks (ANNs) are a cornerstone of modern artificial intelligence, drawing inspiration from the biological neural networks that make up animal brains [32–34]. An ANN is composed of layers of interconnected nodes or neurons, which collectively process information through their interconnections. These networks are highly adept at tasks involving pattern recognition, prediction, and classification, making them essential in diverse applications such as image recognition, natural language processing, and financial forecasting. Advanced techniques, like prompt engineering, further enhance these networks' capabilities, enabling significant improvements in complex tasks such as multi-class classification [35]. The basic structure of an ANN includes three primary layers: the input layer, one or more hidden layers, and the output layer. The input layer receives the data, which is then processed through successive hidden layers that extract features and patterns. Each neuron in these layers applies a transformation function to the inputs it receives, often using a non-linear activation function like Rectified Linear Unit (ReLU) or Sigmoid. This transformation helps the network learn complex patterns. The final layer, or the output layer, produces the results based on the learned patterns.

Training an ANN involves adjusting the weights of the connections between neurons [36,37]. This is typically done using a method known as backpropagation, coupled with an optimization algorithm like stochastic gradient descent. Backpropagation effectively adjusts the weights by calculating the gradient of the network's error with respect to each weight. It then updates the weights to minimize the error, improving the model's predictions. The ability of ANNs to learn from large datasets and improve their accuracy over time without explicit programming for the task is a significant advantage. They are capable of generalizing from the data they train on, which enables them to perform well on new, unseen data, assuming the new data resemble the training data.

The neural network used in this study is structured with multiple dense layers, each employing a ReLU activation function [38,39] to introduce non-linearity and facilitate complex pattern learning within

the data. Starting with a dense layer of 256 neurons, the network architecture gradually reduces the dimensionality through successive layers—128, 64, 32, 16, and 8 neurons—before concluding with a layer of 4 neurons, also using ReLU. The final layer consists of 2 neurons with a softmax activation function, designed to output the probabilities for the two classes: benign and malicious.

Domain adaptation

Domain Adaptation is a technique in machine learning that aims to adapt a model trained on a source domain (where abundant labeled data is available) to perform well on a different but related target domain (where labeled data is scarce or unavailable). This approach is crucial in scenarios where the distribution of data in the training set (source domain) differs from the data in the deployment environment (target domain), which can significantly degrade the performance of the model.

Correlation Alignment (CORAL) shown in equation (1) is a specific method used in domain adaptation to minimize domain shift by aligning the second-order statistics (covariances) of the source and target domain feature distributions. This is achieved by adjusting the feature distributions of both domains so that their covariances are similar, reducing the domain discrepancy at the feature level. Building on the concept of CORAL, DeepCORAL integrates this alignment directly into the training of deep neural networks. DeepCORAL adds a loss term that minimizes the difference in covariances between the last layer features of the source and target domains during the training process. This approach enables the network to learn features that are more invariant to the change in domains, thus improving the model's ability to generalize from the source to the target domain.

$$\text{CORAL Loss} = \frac{1}{4d^2}|C_s - C_T|_F^2 \tag{1}$$

Where, C_s and C_T are the covariance matrices of the source and target domain features, respectively. $|C_s - C_T|_F^2$ represents the Frobenius norm of the difference between the source and target covariance matrices. The Frobenius norm is used to measure the 'distance' between these matrices. is the dimensionality of the output features from the layers where the CORAL loss is being applied. The factor $\frac{1}{4d^2}$ is a normalization term that helps in controlling the scale of the loss relative to other terms in the network's loss function.

In our implementation, we adopt the DeepCORAL approach, utilizing the CORAL metric to compute the distribution differences between the source and target domains. The network architecture used for the target domain mirrors that of the source domain described earlier, featuring the same sequence of dense layers with ReLU activations—256, 128, 64, 32, 16, and 8 neurons. However, it omits the final output layer because the target domain data is treated as unlabeled during training.

Implementation details

The model's trainable parameters were optimized using the Adaptive Moment Estimation (Adam) optimizer [40,41], with a learning rate set at 0.001. The training process was configured with 1000 epochs and a batch size of 512. All training sessions were conducted using the TensorFlow framework with RTX 3090 graphics card.

4. Results and discussion

4.1 The performance of the model

Figure 5 and **Table 1** visualizes the performance of five machine learning models (ANN, LR, DT, KNN, RF) widely used in many tasks [42–44] across three different scenarios: direct prediction in the source domain, direct prediction in the target domain, and domain adaptation in the target domain. The accuracies of each model under these conditions are depicted clearly. Due to the inability of other models to optimize distribution differences directly like DeepCORAL, for models other than ANN, we first applied CORAL to transform the data before proceeding with predictions.

Key observations from these results include: (1) Performance in Source Domain: The ANN model exhibits the highest accuracy (0.90), followed by the RF model at 0.87, showcasing strong performance when the models operate within the domain of their training data. (2) Performance Drop in Target Domain: A significant decline in performance is observed for all models when directly predicting in the target domain without adaptation. For instance, the accuracy of the ANN model drops from 0.90 in the source domain to 0.64 in the target domain. (3) Enhanced Performance with Domain Adaptation: When domain adaptation techniques are applied, there is a marked improvement in model accuracy in the target domain. The ANN model's accuracy improves from 0.64 to 0.81, illustrating the effectiveness of domain adaptation in bridging the gap between different data distributions. From these observations, the domain adaptation can significantly boost the performance of all models in the target domain. This confirms the value of employing domain adaptation strategies, especially in situations where models are expected to function in environments different from their training context.

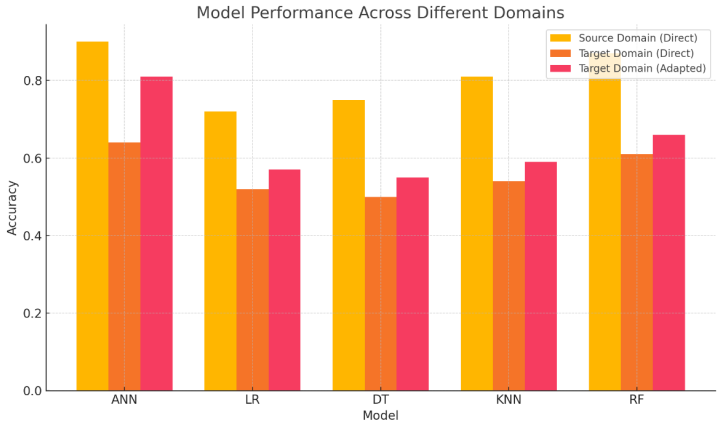

Figure 5. The performance comparison of different models.

Table 1. Detailed numerical prediction performance of different models.

Model Name	Testing data in Source Domain (Direct prediction)	Testing data in Target Domain (Direct prediction)	Testing data in Target Domain (Domain adaptation)
ANN	0.90	0.64	0.81
LR	0.72	0.52	0.57
DT	0.75	0.50	0.55
KNN	0.81	0.54	0.59
RF	0.87	0.61	0.66

Figure 6 shows the accuracy of domain adaptation in an ANN at different layers, each varying in the number of neurons. Starting with 128 neurons and progressing down to 4, the accuracy on the target domain is tracked to evaluate the efficacy of domain adaptation applied via CORAL alignment at each layer. As observed, the accuracy increases significantly as the number of neurons in the layers decreases, starting from 0.66 with 128 neurons and peaking at 0.81 with 4 neurons. The mid-layer with 32 neurons showed a dip to 0.69 before climbing again, indicating an interesting fluctuation in performance.

The performance variations across different layers can be attributed to the varying degrees of feature abstraction at each layer. Deeper layers (with fewer neurons) in an ANN tend to capture more abstract representations of the data [45,46]. When CORAL [47,48] is applied to these layers, it aligns these high-level features more effectively between the source and target domains, leading to improved accuracy. Layers with more neurons capture more granular, less

abstract features. While alignment at these layers can be beneficial, the high dimensionality and complexity might make it harder for CORAL to effectively reduce domain discrepancy at this level, resulting in lower accuracy. The dip at 32 neurons and subsequent rise as layers become deeper suggest that there might be an optimal point of abstraction for domain adaptation, where the features are neither too granular nor too abstract, balancing detail with domain-invariant representation.

Figure 7 illustrates the impact of varying CORAL coefficient values on the accuracy of a model in domain adaptation scenarios. The coefficients tested range from 0.01 to 0.5, and the corresponding accuracies are plotted to show how adjustments in the CORAL coefficient influence model performance. A clear trend is observed where the accuracy increases with the CORAL coefficient from 0.01 to 0.2, reaching a peak accuracy of 0.81 at a coefficient of 0.2. Beyond this point, at a coefficient of 0.5, the accuracy declines to 0.77, suggesting that too strong a weighting on the CORAL loss relative to the classification loss might be detrimental.

The initial increase in accuracy with higher CORAL coefficients suggests that stronger alignment of feature distributions between the source and target domains is beneficial up to a certain point. This aligns with the concept that reducing domain discrepancy helps the model generalize better to the target domain. The decrease in accuracy at the highest coefficient (0.5) indicates an overemphasis on domain alignment, potentially at the cost of losing relevant information necessary for accurate predictions. This could lead to the model overfitting to the domain adaptation criterion rather than focusing on the predictive task.

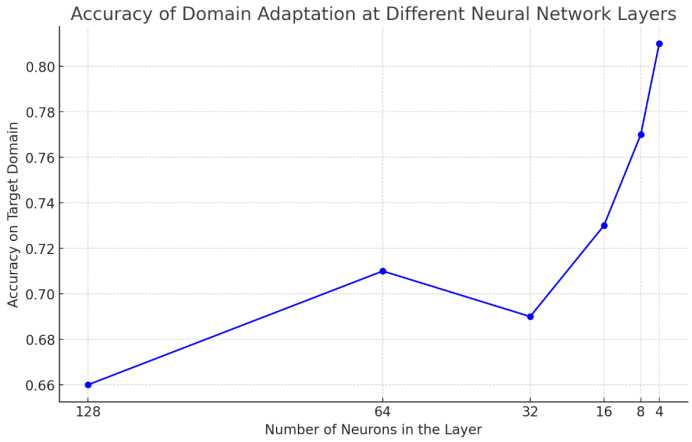

Figure 6. The performance based on different layers of ANN.

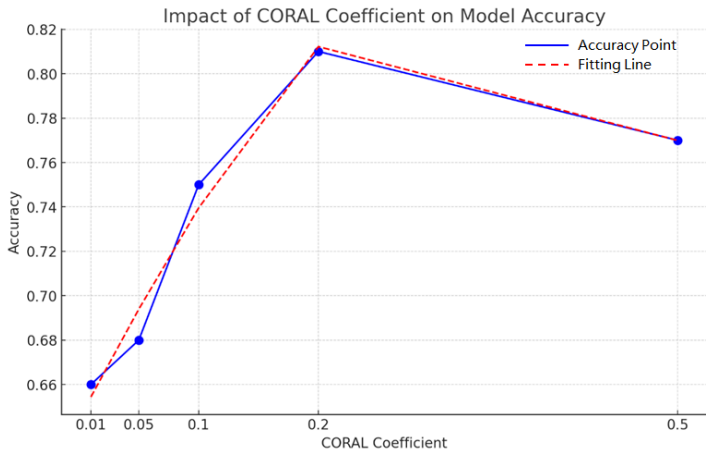

Figure 7. The influence of coral coefficients on the model accuracy.

4.2 Discussion

Domain adaptation, especially through methods like CORAL, effectively bridges the gap between source and target domain distributions, as seen in the substantial accuracy improvements from direct predictions in the target domain to those with domain adaptation. For instance, the ANN model shows a notable increase from 0.64 to 0.81 accuracy, demonstrating domain adaptation's potential to mitigate the challenges posed by domain shift. While domain adaptation offers considerable benefits, it is not without its limitations. The variance in performance improvement across different models and the dependency on hyperparameter tuning (such as the CORAL coefficient) indicate areas that could be improved. For example, excessive emphasis on the CORAL loss, as evidenced by the decrease in accuracy at higher coefficients, suggests a delicate balance is required to optimize both domain adaptation and classification performance. Furthermore, the fluctuating performance across different layers of the ANN when applying CORAL suggests that the effectiveness of domain adaptation may depend significantly on the nature and depth of the layers involved. This observation points to the potential need for layer-specific adaptation strategies, where domain adaptation parameters are tailored based on the layer's characteristics and its role in the network.

Future studies could explore adaptive mechanisms that dynamically adjust domain adaptation [49,50] parameters during training, potentially guided by real-time feedback on model performance in both the source and target domains. Additionally, investigating domain adaptation across more diverse and complex datasets could further validate the robustness and versatility of these techniques [51].

5. Conclusion

The results of this study underscore the effectiveness of domain adaptation in addressing the challenges posed by the dynamic nature of Android malware. Traditional malware detection methods often fail to generalize well to new, unseen environments, leading to decreased performance and increased risk. By implementing domain adaptation techniques, this research demonstrates a notable improvement in model accuracy across varied data distributions, particularly in environments different from the training context. For instance, the application of CORAL-based domain adaptation methods allowed for substantial gains in accuracy, highlighting the technique's ability to mitigate the adverse effects of domain shifts.

Moving forward, future research should focus on developing more adaptive domain adaptation mechanisms that could dynamically adjust based on real-time model performance feedback. Additionally, exploring these strategies across a broader range of datasets and real-world scenarios will be crucial to fully ascertain the robustness and applicability of domain adaptation in malware detection.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (2021YFB31012003、2021YFB31012004、2021YFB31012005)

References

[1] K. Liu, S. Xu, G. Xu, et al., 2020. A review of android malware detection approaches based on machine learning. *IEEE access.* 8, 124579–124607.

[2] Y. Zhou, X. Jiang, 2012. Dissecting android malware: Characterization and evolution. *2012 IEEE symposium on security and privacy*. IEEE. pp. 95–109.

[3] P. Ren, Z. Zhao, 2024. Parental recognition of double reduction policy, family economic status and educational anxiety: exploring the mediating influence of educational technology substitutive resource. *Economics and Management Information.* 1–12.

[4] F. Yu, J. Milord, S. L. Orton, et al., 2022. The concerns and perceived challenges students faced when traditional in-person engineering

courses suddenly transitioned to remote learning. *2022 ASEE Annual Conference*.

[5] P. R. Pardhi, J. K. Rout, N. K. Ray, 2021. Implementation of a malware scanner using signature-based approach for android applications. *2021 19th OITS International Conference on Information Technology (OCIT)*. IEEE. pp. 14–19.

[6] L. Zhou, Z. Luo, X. Pan, 2024. Machine learning-based system reliability analysis with Gaussian Process Regression. *arXiv preprint arXiv:2403.11125*.

[7] Y. Liu, L. Liu, L. Yang, et al., 2021. Measuring distance using ultra-wideband radio technology enhanced by extreme gradient boosting decision tree (XGBoost). *Automation in Construction.* 126, 103678.

[8] H. Wang, Y. Zhou, E. Perez, et al., 2024. Jointly Learning Selection Matrices For Transmitters, Receivers And Fourier Coefficients In Multichannel Imaging. *arXiv preprint arXiv:2402.19023*.

[9] Y. Liu, Y. Bao, 2023. Real-time remote measurement of distance using ultra-wideband (UWB) sensors," *Automation in Construction.* 150, 104849.

[10] Y. Qiu, J. Wang, 2024. A Machine Learning Approach to Credit Card Customer Segmentation for Economic Stability. *Proceedings of the 4th International Conference on Economic Management and Big Data Applications, ICEMBDA, October 27–29, 2023, Tianjin, China.*

[11] M. Li, J. He, G. Jiang, et al., 2024. DDN-SLAM: Real-time dense dynamic neural implicit SLAM with joint semantic encoding. *arXiv preprint arXiv:2401.01545*.

[12] Y. Qiu, 2019. Estimation of tail risk measures in finance: Approaches to extreme value mixture modeling. Johns Hopkins University.

[13] Y. Zhou *et al.*, 2023. Semantic Wireframe Detection.

[14] F. Zhao, F. Yu, T. Trull, et al., 2023. A new method using LLMs for keypoints generation in qualitative data analysis. *2023 IEEE Conference on Artificial Intelligence (CAI)*. IEEE. pp. 333–334.

[15] S. Li, K. Singh, N. Riedel, et al., 2022. Digital learning experience design and research of a self-paced online course for risk-based inspection of food imports. *Food Control.* 135, 108698.

[16] Y. Liu, H. Yang, C. Wu, 2023. Unveiling patterns: A study on semi-supervised classification of strip surface defects. *IEEE Access.* 11, 119933–119946.

[17] A. Farahani, S. Voghoei, K. Rasheed, et al., 2021. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020.* 877–894.

[18] G. Csurka, 2017. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*.

[19] S. Ben-David, J. Blitzer, K. Crammer, et al., 2006. Analysis of representations for domain adaptation. *Advances in neural information processing systems.* 19.

[20] Y. Qiu *et al.*, 2024. A novel image expression-driven modeling strategy for coke quality prediction in the smart cokemaking process. *Energy.* 294, 130866.

[21] Y. Hao, Z. Chen, J. Jin, et al., 2023. Joint operation planning of drivers and trucks for semi-autonomous truck platooning. *Transportmetrica A: Transport Science.* 1–37.

[22] S. Xiong, H. Zhang, M. Wang, et al., 2022. Distributed Data Parallel Acceleration-Based Generative Adversarial Network for Fingerprint Generation. *Innovations in Applied Engineering and Technology.* 1–12.

[23] J. Lee, H. Jang, S. Ha, et al., 2021. Android malware detection using machine learning with feature selection based on the genetic algorithm. *Mathematics.* 9(21), 2813.

[24] C. Palma, A. Ferreira, M. Figueiredo, et al., 2024. Explainable machine learning for malware detection on android applications. *Information.* 15(1), 25.

[25] J. Senanayake, H. Kalutarage, M.O. Al-Kadri, 2021. Android mobile malware detection using machine learning: A systematic review. *Electronics*. 10(13), 1606.

[26] X. Pan, Z. Luo, L. Zhou, 2024. Navigating the landscape of distributed file systems: Architectures, implementations, and considerations. *arXiv preprint arXiv:2403.15701*.

[27] X. Deng, L. Li, M. Enomoto, aet al., 2019. Continuously frequency-tuneable plasmonic structures for terahertz bio-sensing and spectroscopy. *Scientific reports*. 9(1), 3498.

[28] X. Deng, Y. Kawano, 2018. Surface plasmon polariton graphene midinfrared photodetector with multifrequency resonance. *Journal of Nanophotonics*. 12(2), 026017–026017.

[29] Kaggle. Network Traffic Android Malware [Internet]. https://www.kaggle.com/datasets/xwolf12/network-traffic-android-malware (cited May 1, 2024).

[30] M. Ahmed, R. Seraj, S.M.S. Islam, 2020. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*. 9(8), 1295.

[31] K. Krishna, M.N. Murty, 1999. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. 29(3), 433–439.

[32] L. Zhou, H. Zhang, N. Zhou, 2024. Double-compressed artificial neural network for efficient model storage in customer churn prediction. *Artificial Intelligence Advances*. 6(1), 1–12.

[33] S. Li, P. Kou, M. Ma, et al., 2024. Application of Semi-supervised Learning in Image Classification: Research on Fusion of Labeled and Unlabeled Data. *IEEE Access*.

[34] F. Chen, Z. Luo, L. Zhou, et al., 2024. Comprehensive survey of model compression and speed up for vision transformers. *arXiv preprint arXiv:2404.10407*.

[35] F. Zhao, F. Yu, 2024. Enhancing multi-class news classification through bert-augmented prompt engineering in large language models: a novel approach. *The 10th International scientific and practical conference "Problems and prospects of modern science and education"(-March 12–15, 2024) Stockholm, Sweden. International Science Group. 381 p.*, 2024, p. 297.

[36] S. Agatonovic-Kustrin, R. Beresford, 2000. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of pharmaceutical and biomedical analysis*. 22(5), 717–727.

[37] D. Jahed Armaghani, M. Hasanipanah, A. Mahdiyar, et al., 2018. Airblast prediction through a hybrid genetic algorithm-ANN model. *Neural Computing and Applications*. 29, 619–629.

[38] C. Banerjee, T. Mukherjee, E. Pasiliao Jr, 2019. An empirical study on generalizations of the ReLU activation function. *Proceedings of the 2019 ACM Southeast Conference*. 164–167.

[39] K. Eckle, J. Schmidt-Hieber, 2019. A comparison of deep networks with ReLU activation function and linear spline-type methods. *Neural Networks*. 110, 232–242.

[40] D. P. Kingma, J. Ba, 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[41] Z. Zhang, 2018. Improved adam optimizer for deep neural networks. *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*. IEEE. pp. 1–2.

[42] Y. Qiu, J. Wang, Z. Jin, et al., 2022. Pose-guided matching based on deep learning for assessing quality of action on rehabilitation training. *Biomedical Signal Processing and Control*. 72, 103323.

[43] S. Orton, F. Yu, L. Flores, et al., 2023. Student perceptions of confidence in learning and teaching before and after teaching improvements. *2023 ASEE Annual Conference*. ASEE PEER.

[44] Y. Wu, Z. Jin, C. Shi, et al., 2024. Research on the application of deep learning-based BERT model in sentiment analysis. *arXiv preprint arXiv:2403.08217*.

[45] Y. Hao, Z. Chen, X. Sun, et al., 2024. Planning

of Truck Platooning for Road-Network Capacitated Vehicle Routing Problem. *arXiv preprint arXiv:2404.13512*.

[46] Y. Liu,Y. Bao, 2021. Review of electromagnetic waves-based distance measurement technologies for remote monitoring of civil engineering structures. *Measurement.* 176, 109193.

[47] B. Sun, K. Saenko, 2016. Deep coral: Correlation alignment for deep domain adaptation. *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer. pp. 443–450.

[48] B. Sun, J. Feng, K. Saenko, 2017. Correlation alignment for unsupervised domain adaptation. *Domain adaptation in computer vision applications.* 153–171.

[49] Z. Cao, L. Ma, M. Long, et al., 2018. Partial adversarial domain adaptation. *Proceedings of the European conference on computer vision (ECCV)*. 135–150.

[50] M. Wang, W. Deng, 2018. Deep visual domain adaptation: A survey. *Neurocomputing.* 312, 135–153.

[51] W. M. Kouw, M. Loog, 2019. A review of domain adaptation without target labels. *IEEE transactions on pattern analysis and machine intelligence.* 43(3), 766–785.