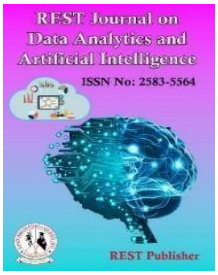
REST Journal on Data Analytics and Artificial Intelligence
Vol: 4(1), March 2025
REST Publisher; ISSN: 2583-5564
Website: http://restpublisher.com/journals/jdaai/
DOI: https://doi.org/10.46632/jdaai/4/1/42



Malware Detection in Android Using Machine Learning

[*]Manoranjan Dash, Vismaya Reddy B, Srikanth V, Venkata Naga Karthik A
Anurag University, Hyderabad, Telangana,India.
[*]Corresponding Author Email: manoranjanai@anurag.edu.in

Abstract. *One item that almost everyone has in their everyday life is a mobile phone, and Android-based phones dominate the market. Numerous apps can be found on the internet, and individuals frequently download applications from unidentified sites that may harm their mobile devices. Therefore, these dangerous apps may risk their money, privacy, and data security. In order to prevent consumers from installing harmful applications and protect their devices, a machine learning model can be used to detect such apps before installation. Given the complexity and dynamic nature of the malwares, traditional security approaches alone may not be suitable enough to protect users effectively. Machine learning (ML) techniques offer a promising avenue for enhancing malware detection and mitigation efforts. We have been motivated to implement various well known ML Algorithms such as Support Vector Machine, Random Forest decision trees, and Naive Bayes. Futher we will also implement Artificial Neural Networks (ANNs) or simple Neural Networks to find the efficacy of the Malware Detection.*

Keywords: *Android Malware Detection, Machine Learning, Feature Categorization, Artificial Neural Network, Static Analysis, Permissions, API Calls, Drebin Dataset.*

1. INTRODUCTION

Mobile devices, particularly those running on the Android operating system, have become ubiquitous today, serving as must-have tools for communication, entertainment, productivity, and much more. With millions of applications available for download from various sources, users enjoy a wide range of functionalities tailored to their needs. However, this vast ecosystem also presents significant challenges in terms of security and privacy. The huge presence of malicious applications, or malware on the internet pose a serious threat to users' data, privacy, and device security [1-5]. Despite efforts to mitigate these risks, many users remain unaware of the potential dangers existing in the digital landscape [6].

One of the primary challenges facing mobile users is the presence of malicious applications, commonly referred to as malware. These applications are designed with malicious intent, aiming to compromise users' devices, steal sensitive information, or engage in fraudulent activities [7]. Malware can take various forms, including spyware, ransomware, adware, and trojans, each posing unique risks to users' privacy and security[8].

Many users may inadvertently download and install malware-infected applications, either due to ignorance or a lack of caution [9]. This unawareness leaves them vulnerable to various threats, including data breaches, financial losses, and identity theft. With the proliferation of online forums, marketplaces, and websites offering applications for download, identifying trustworthy sources has become increasingly challenging. Malicious actors often disguise their malware as legitimate applications, making it difficult for users to distinguish between safe and harmful software.

While mobile operating systems such as Android incorporate security features and mechanisms to detect and

prevent malware, these measures are not foolproof. Malware authors continuously evolve their tactics to evade detection and circumvent security protocols, posing a persistent threat to users' devices and data. Moreover, users may fail to update their devices or install security patches promptly, further exposing them to potential risks.

Given the complexity and dynamic nature of the malware landscape, traditional security approaches alone may not suffice to protect users effectively. Machine learning (ML) techniques offer a promising avenue for enhancing malware detection and mitigation efforts. By leveraging ML algorithms trained on large datasets of known malware samples, researchers and security professionals can develop robust models capable of identifying suspicious patterns and behaviors indicating malicious activity.

Artificial Neural Networks (ANNs) or simple Neural Networks present a significant advancement in the field of malware detection compared to traditional regression and decision tree models. While conventional methods may struggle to keep up with the evolving sophistication of malware, ANNs offer a superior solution. Their ability to handle complex, nonlinear relationships within data allows them to capture intricate patterns and behaviors associated with malware effectively [10-13].

Further a feature type-based training is implemented. By doing so, it allows for more targeted and specialized detection strategies, potentially improving the overall accuracy and effectiveness of malware detection systems. This approach represents a departure from traditional methods that treat all features uniformly and demonstrates an innovative way to leverage domain knowledge for better model performance [14-16].

A. Objective

This project aims to perform a static malware analysis on APKs data based on the permissions required by the application using ANN model. The project also includes a Feature Group-Based Malware Detection. This approach provides insights into how different types of features, such as API call signatures, manifest permissions, and intents, contribute to the accuracy of the model in detecting malware.

B. Approach

The initial phase of the project involved collecting a dataset containing information about various Android applications. The dataset was obtained from reliable sources and comprised features related to API call signatures, manifest permissions, intents, and other relevant attributes. An exploratory data analysis was conducted to gain insights into the dataset's structure and contents. approximately 11,500 new Android malware instances emerging on a daily basis. [3]

According to Kaspersky security, in quarter one of 2022, mobile malware, adware, and riskware exceeded 6.4 million attacks. Malicious installation packages were more than 500k, among which 53,947 packages were related to mobilebanking Trojans, and 1,942 packages were mobile ransomware. About half of these detected threats were unwanted riskTool apps followed by Adware apps.[4]

B. Artificial Neural Networks For Machine Learning

Data pre-processing involved cleaning the data by handling missing values, removing duplicates, and addressing any inconsistencies in the dataset. Categorical variables were encoded using appropriate techniques, while numerical features were normalized to a standard scale. Additionally, the dataset was partitioned into subsets based on distinct feature categories, such as API call signatures, manifest permissions, and intents.

The subsequent step in the project involved developing an neural network model using the Keras deep learning framework. The archcitecture, optimizers and activation functions are included. Multiple instances of the model were trained, with each instance using a different subset of features corresponding to a specific feature category.

To assess the performance of the trained models, the dataset was split into separate training and testing sets. The models were then trained on the training data and evaluated on the testing data to measure their classification accuracy. This process was repeated for each model variant trained on different feature categories. The accuracy results obtained from each model variant were collected and analyzed to determine the efficacy of individual feature categories in

distinguishing between benign and malicious applications [17-20].

Following model evaluation, the accuracy of the models trained on individual feature categories was compared with the accuracy of the model trained on the entire dataset. By analyzing the performance of various feature subsets, insights were gained into the importance of different feature categories for malware detection in Android applications.

2. BACKGROUND

A. Malware In Android

Android as an operating system in mobiles has rapidly grown over the years. The Android operating system, due to its open-source nature and popularity, has become a prime target for malware developers seeking to exploit vulnerabilities and gain unauthorized access to users' devices and data. Malware in the Android ecosystem encompasses a wide range of threats, including viruses, trojans, adware, spyware, and ransomware. As per (Statista, 2021), the total number of new Android malware samples till March 2020 amounted to 482,579 per month. Additionally, G DATA Cyber Defence AG (Burris, 2020) reported that over 4.18 million malicious applications were found on the Android platform in 2019, with Artificial Neural Networks (ANNs) represent a class of machine learning models inspired by the structure and function of biological neural networks in the human brain.

ANNs consist of interconnected nodes, or neurons, organized into layers, including input, hidden, and output layers. These networks are capable of learning complex patterns and relationships in data through a process known as training, where the network adjusts its parameters based on observed input-output pairs. Most machine learning today is Artificial Neural Networks. Because of the recent rise in computing power, these Neural Networks have turn into tremendously popular, and they are presently found practically everywhere. The neural networks provide the intellectual edge that retains us involved in each application in the real world. ANNs have vast range of applications such as health care and diagnosis, facial recognition, stock market forecasts, defense. [5]

ANNs started as perceptrons consisting multiple layers of connected nodes which late developed into complex architecture styles such as recurrent neural network (RNN) and convolutional neural network (CNN). For a simple nueral network consisting of one hidden layer and one output neuron, the computation formula is as

$$\text{Output} = f_{\text{o}}\left[\text{BO} + \sum_K W_K f_{\text{H}}\left(\text{BH}_K + \sum_i W_i \text{Input}_i\right)\right]$$

Equation. 1. Mathematical Formula for single layer perceptron

where W is the weight between nodes, B represent the biases, and *f* represents the activation functions. [8]

3. RELATED WORK

A. Existing Research

In *Permission-Based Malware Detection in Android Using Machine Learning* [1], This research focuses on developing an effective Android malware detection system by analyzing app permissions and using machine learning techniques to classify apps as benign or malicious. The study collected a dataset of 10,000 apps, extracting features like permissions, app size, and permission rates. Various machine learning algorithms, including Support Vector Machine (SVM), Random Forest, and Naive Bayes, were employed for classification. The proposed approach improved detection accuracy by optimizing the permission feature set, achieving a detection accuracy of up to 97%. The research highlights the importance of permission- based analysis in malware detection, offering a lightweight and efficient solution for identifying malicious Android applications.

In *Permission-Based Android Malware Detection by Zarni Aung, Win Zaw* [2], the authors have demonstrated machine learning based framework to detect malware in android applications. They have extracted permission feature requests from the applications they have used in their dataset and stored them as 0 or 1. K- means algorithm, random forest, and decision trees have been implemented in this work with the Random Forest model showing best accuracy of 92%.

Machine learning models have been presented in multiple works as an effective solution to detect malware in APK files. In *Detection of Android Malware using Machine Learning* [7], various features such as manifest permission, intents, system commands, and API calls have been extracted from APK files and used a large dataset compared to other works. They have achieved an accuracy of 98% stating it as higher than other works. The work has used machine learning models such as J48 classifier, Random forest classifier, KNN, SVM, Naïve Bayes, MultiLayer Perceptron, and Decision Table.

In *An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms* [8], the authors have used the CICInvesAndMal2019 dataset. They have used the Naïve Bayes, SVM and KNN classifiers on permission features and API calls separately with the SVM model providing highest accuracy of 94%.

Android malware detection involves two strategies – permission based or static analysis and dynamic analysis. Static features are obtained by analyzing the source code or other information associated with the application are called static features. [9] Files including AndroidManifest.xml, smali files, etc., can be obtained by decompiling APK files. Further analysis of these files reveals a set of static features, including permissions, API calls, Dalvik opcodes, and other components. [10]. When running Android applications in real environments or emulation environments such as a sandbox, the acquired runtime behavioral features are known as dynamic features.[11] The features in dynamic analysis include system calls, API calls, network traffic, and CPU data [10].

B. Research Gaps

Several gaps exist in the current landscape of malware detection for Android devices. One significant gap is the lack of feature-centric analysis—many systems fail to assess the individual impact of different feature subsets, such as API calls and permissions, which could enhance detection efficiency. Furthermore, limited generalization remains a challenge, with most models unable to perform consistently across varied datasets. Another issue is the absence of feature engineering and optimization, which results in models that are not well-suited to evolving malware patterns. Scalability is another area of concern, as many systems are not designed to handle real-world, large- scale applications effectively. Moreover, inadequate robustness testing makes it difficult to trust these models for deployment in practical scenarios.

4. METHODOLOGY

The steps followed in this work include -

a. Collecting the dataset
b. Data pre-processing
c. Creating data subsets based on types of features categories.
d. Defining model architecture of ANN
e. Data splitting and Encoding
f. Training the model with multiple subsets of data
g. Comparison of accuracies of models with different subsets

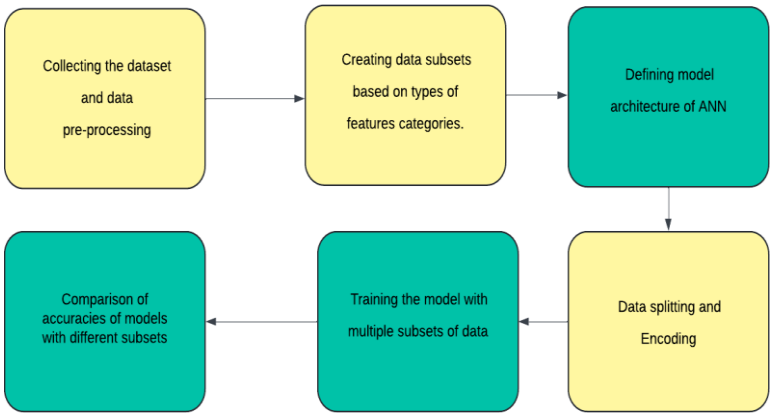


FIGURE 1. High Level Design of the project

As defined in the problem statement, along with usage of ANN to predict malware in APK files, additionally the work also aims to detect the impact or dominance of each feature category when used individually in detection. This could be a valuable factor in building much secure detections systems.

The method defined for the process of data splitting, model training and evaluation is used for multiple calls. The method is called for data subsets for the two majority categories of API call signature, Manifest permission. Further each of these two are also combined with intents and used in predictions. Another combination of these two subsets is also tested and atlast the entire data subset is used in model training and prediction.

All of the accuracy values returned in each method call are accumulated for their respective category. This disctionary is used to analyse the results and a bar graph is plotted for visual representation of the results.

DATASET

The dataset used is the popular Drebin dataset of android APK features. The data consists of two files features-categories and features values. The features categories consist of different types of static features and their respective categories. There are a total of 215 features and 4 categories namely - 'API call signature', 'Manifest Permission', 'Intent', and 'Commands signature'. The features values is a .csv file consists of 216 columns representing each feature and rows representing 0s or 1s indicating presence of the feature in the application file.

The last column represents the class variable with B standing for benign or safe application and S representing Malware classes. There are total of 5560 malware records and 9476 benign records representing APKs.

.

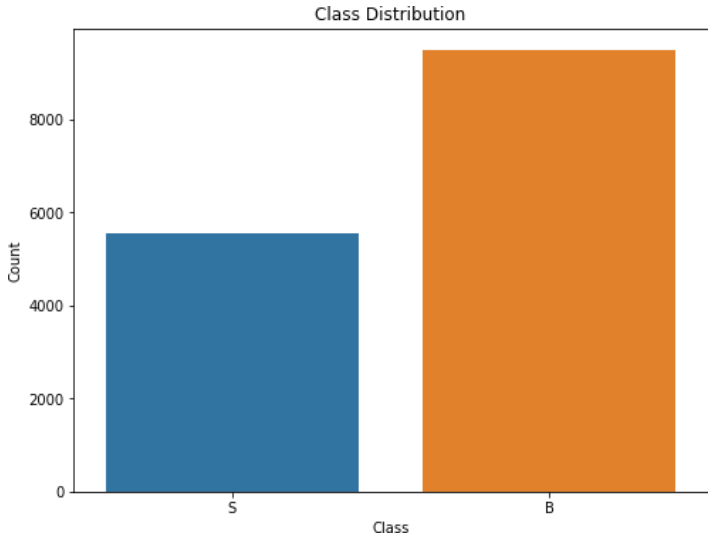


FIGURE 2. Classes in the dataset

IMPLEMENTATION

A. Data Exploration and Preprocessing

The features-categories file consists of 215 records with two columns – transact and Api call signature representing feature and its category. There are no null values present in the data. The other file consists of 216 columns of each feature of total of 15036 records – APKs data. This file also is free from null values. There are few records which contain '?' symbol in records which are replaced by null values and are eliminated.

The further step was to create the data subsets based on feature categories. Firstly, based on the feature-category data a dictionary of category and its related features are grouped. Based on this dictionary, a new dictionary of data subsets is created consisting of key as the category and values being the values of 0s and 1s for the feature belonging to that

category. The respective class values are also appended to these records. The obtained categories dictionary includes 5 types of features with API call signature having 72 values, Manifest Permission with highest of 113 values, and 23 intents and 6 command signatures.

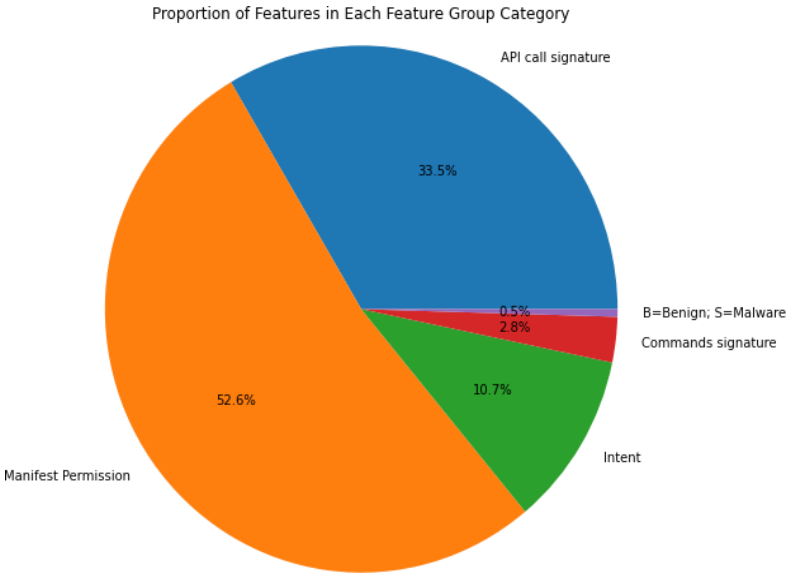


FIGURE 3. Distribution of feature categories

B. Model Architecture and Data Training

A function is defined to take input of data collection of records of 0s and 1s of API features, and its class values. Data subsets created would be passed to this function. A deep copy of the data is created and class values are encoded using label Encoder. The X and y variables are defined as the features and class values, which are then split into training and test subsets with a test data size of 20%.

Further, using the tensorflow.keras library ANN model architecture is written. A sequential model is created with 2 dense layers, 2 dropout layers and one output layers. The input size is 128 and activation function used is Relu . Relu activation function ensures positive output from the values. Dropout of value 0.3 is applied on it and passed to another dense layer of size 64. Another dropout layer is applied and finally an output layer is used with sigmoid activation function due to binary classification task requirement. The model would then be compiled and fit on training data with loss function as binary cross entropy again attributing to the binary class nature, optimizer as Adam, and metrics as accuracy. The model is fit on the train data with a validation set of 0.1 size in batch size of 32 for 10 epochs. The model is then used to predict on the test data and accuracy of these predictions is returned as the result of this function.

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_33 (Dense)            (None, 128)               27648

 dropout_22 (Dropout)        (None, 128)               0

 dense_34 (Dense)            (None, 64)                8256

 dropout_23 (Dropout)        (None, 64)                0

 dense_35 (Dense)            (None, 1)                 65

=================================================================
Total params: 35,969
Trainable params: 35,969
Non-trainable params: 0
```

FIGURE 4. Model Architecture

5. RESULTS

The result set dictionary of all data subsets used and their respective accuracies is – {'API call signature': 0.97904888593282 35, 'Manifest Permission': 0.967741935483871, 'API call signature and Intents': 0.62 46465990354233, 'Manifest Permission and Intents': 0.6 219856976550806, 'API call signature and Manifest Permi ssion': 0.62780641942458, 'All Classes': 0.9880279348187563}

The results present intriguing insights into the importance of different feature categories in the model. While the overall performance on the entire dataset was expected to be high, other feature subsets yielded contrasting results. Training the model on the complete dataset achieved a remarkable accuracy of approximately 98%.

When trained on the feature category with the highest number of features, namely "Manifest permission," the model achieved an accuracy of 96%. This underscores the importance of this category in understanding the nature of an application. Similarly, the "API call signature" category demonstrated an even higher accuracy, reaching up to 97%, suggesting its critical role in determining an application's behavior.

Surprisingly, appending the "Intents" category, the third- largest category, to the subsets resulted in significantly lower accuracies of around 62%. This indicates that including intents as additional context does not necessarily guarantee higher accuracy and may not sufficiently characterize an application's behavior.

Further experimentation with combined feature categories, such as appending manifest permissions with API call signatures, also yielded low accuracy scores of 62%, highlighting the importance of feature engineering in this task and its significant impact on system performance.

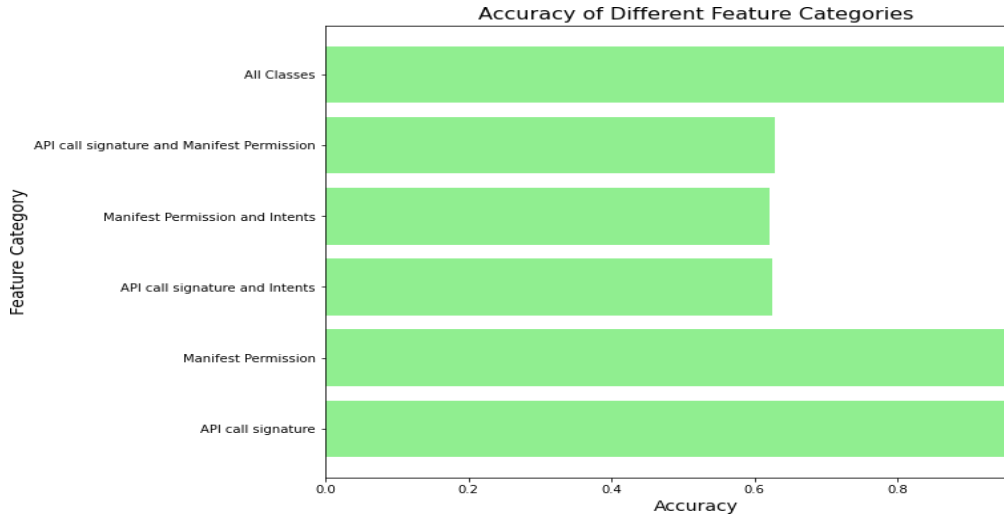


FIGURE 6. Bar graph representing result values and category used.

	accuracy	precision	Recall	F1
API call signature	0.977386	0.979497	0.958067	0.9
Manifest Permission	0.959428	0.966024	0.925242	0.9
API call signature and Intents	0.623150	0.000000	0.000000	0.0
Manifest Permission and Intents	0.634958	0.000000	0.000000	0.0
API call signature and Manifest Permission	0.638117	0.000000	0.000000	0.0
All Classes	0.989358	0.983245	0.988475	0.9

FIGURE 7. Classification Metrics

These three categories notably gave precision and recall values of 0 indicating they are predicting all data inputs as non-malware or benign samples. It suggests the inability of model to predict malware on these combinations of data. There might not be any distuinguishable patterns identified by the model from the given subsets. This might also be due to limited samples of the malware class in the data.

In conclusion, it is observed that manifest permissions and API call signatures individually contribute substantially to the detection of malware in application files. However, their effectiveness diminishes when combined with other categories, such as intents. This underscores the importance of feature selection and underscores that even individual categories, such as manifest permissions or API call signatures alone, are sufficient to build robust detection models. Additionally, the architecture of the Artificial Neural Network (ANN) demonstrates exceptional efficiency in detecting malicious APKs, achieving a high accuracy of 98%, emphasizing the superiority of neural networks over traditional machine learning models.

6. CONCLUSION AND FUTURE WORK

In this work, the problem of malware in android systems and detection mechanisms have been discussed. This work proposes a new Artificial Neural Network model as a solution to detect APKs using static analysis methodology. Static or feature based analysis provides the advantage of prediction on files without installing them. Additionally, another aspect of malware detection has been discussed based on feature category-based classification and performance. This targeted approach allows for more specialized detection strategies tailored to the unique properties of each feature group. The dataset of combination of all features has been divided into multiple subsets based on category groups. Each of these groups are trained on the proposed ANN model after pre-processing, and splitting and the accuracies are collected. It is observed that, even individual categories prove to be efficient detectors of malware with accuracies around 96%. It is also to be noted that not all combinations of groups can always give good accuracies and knowing feature importance is essential. These results can be essential when there is a limitation onf features or also to discuss upon the dominance of types of feature in the detection of android malware. The entire dataset when used for prediction gave an accuracy of 98% indicating strong ability of the ANN model.

Conclusion:

This study proposes the Artificial Neural Network (ANN) as a robust model for detecting Android malware compared to traditional machine learning algorithms and a new set of inferences based on feature type-based classification. Through the fusion of category-based classification and ANN technology, this research sets a foundation for more sophisticated Android malware detection methods, ultimately enhancing cybersecurity for users worldwide. The performance analysis of feature groups provides valuable insights, with certain categories demonstrating criticality in detecting malware behavior. The study's findings can be instrumental in scenarios with limited feature data and can contribute to discussions on the dominance of feature types in Android malware detection. The inclusion of group-based classification and the utilization of ANN represent novel aspects of this work and contribute significantly to the research in this field.

Future Work:

While this project provides quite important outcomes, there are few areas which can be improved or added as a new dimension to the scope of this work. Including the current feature types, more different features such as user behaviour patterns, and network traffic anlaysis can be included in the future. This work is based on static analysis and thus the dynamic analysis methodology also can be explored in addition to it such as runtime monitoring, sandboxing. Dynamic analysis can provide more real time information and valuable insights. The scale of the data used, and model architecture can be increased to be applicable to realtime situations. Other types of neural networks can also be explored to test in this area such as CNNs or RNNs which might result more efficient results.

REFERENCES

[1]. Kumar, R.A., Mallikarjuna Reddy, A., Chandrasekhar Reddy, T., Ravi Kishore, M. A study of block chain technology and cryptocurrency Journal of Advanced Research in Dynamical and Control Systems, 2018, 10(11 SpecilIssue), pp. 994–1000

[2]. Dayaker, P., Honey Diana, P., Chandrasekhara Reddy, T., Mallikarjuna Reddyreddy, A. Advancements of security and privacy of sensitive data cloud computing, Journal of Advanced Research in Dynamical and Control Systems, 2018, 10(11 SpecilIssue), pp. 956–964

[3]. Mallikarjuna Reddy, A., Venkata Krishna, V., Sumalatha, L. Eficient face recognition by compact symmetric elliptical texture matrix (Csetm), Journal of Advanced Research in Dynamical and Control Systems, 2018, 10(4), pp. 428–439

[4]. Reddy, A.M., Venkata Krishna, V., Sumalatha, L. Face recognition based on stable uniform patterns, International Journal of Engineering and Technology(UAE), 2018, 7(2), pp. 626–634

[5]. Reddy, A.M., Krishna, V.V., Sumalatha, L., Niranjan, S.K. Facial recognition based on straight angle fuzzy texture unit matrix, Proceedings of the 2017 International Conference On Big Data Analytics and Computational Intelligence, ICBDACI2017, 2017, pp. 366–372, 8070865

[6]. Reddy, A.M., Subbareddy, K., Krishna, V.V. Classification of child and adulthood using GLCM based on diagonal LBP, Proceedings of the 2015 International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2015, 2016, pp. 857–861, 7457003.

[7]. Vijay Kumar, V., Ganapathi Raju, N.V., Mallikarjuna Reddy, A. Histograms of term weight feature (HTWF) model for authorship attribution, International Journal of Applied Engineering Research, 2015, 10(16), pp. 37527–37533.

[8]. S. K.Sarangi ,R.Panda & Manoranjan Dash," Design of 1-D and 2-D recursive filters using crossover bacterial foraging and cuckoo search techniques", Engineering Applications of Artificial Intelligence, Elsevier Science,vol.34, pp.109-121,May 2014.

[9]. Manoranjan Dash, N.D. Londhe, S. Ghosh, et al., "Hybrid Seeker Optimization Algorithm-based Accurate Image Clustering for Automatic Psoriasis Lesion Detection", Artificial Intelligence for Healthcare (Taylor & Francis), 2022, ISBN: 9781003241409

[10]. Manoranjan Dash, Design of Finite Impulse Response Filters Using Evolutionary Techniques - An Efficient Computation, ICTACT Journal on Communication Technology, March 2020, Volume: 11, Issue: 01

[11]. Manoranjan Dash, "Modified VGG-16 model for COVID-19 chest X-ray images: optimal binary severity assessment," International Journal of Data Mining and Bioinformatics, vol. 1, no. 1, Jan. 2025, doi: 10.1504/ijdmb.2025.10065665.

[12]. Manoranjan Dash et al.," Effective Automated Medical Image Segmentation Using Hybrid Computational Intelligence Technique", Blockchain and IoT Based Smart Healthcare Systems, Bentham Science Publishers, Pp. 174-182,2024

[13]. Manoranjan Dash et al.," Detection of Psychological Stability Status Using Machine Learning Algorithms", International Conference on Intelligent Systems and Machine Learning, Springer Nature Switzerland, Pp.44-51, 2022.

[14]. Samriya, J. K., Chakraborty, C., Sharma, A., Kumar, M., & Ramakuri, S. K. (2023). Adversarial ML-based secured cloud architecture for consumer Internet of Things of smart healthcare. IEEE Transactions on Consumer Electronics, 70(1), 2058-2065.

[15]. Ramakuri, S. K., Prasad, M., Sathiyanarayanan, M., Harika, K., Rohit, K., & Jaina, G. (2025). 6 Smart Paralysis. Smart Devices for Medical 4.0 Technologies, 112.

[16]. Kumar, R.S., Nalamachu, A., Burhan, S.W., Reddy, V.S. (2024). A Considerative Analysis of the Current Classification and Application Trends of Brain–Computer Interface. In: Kumar Jain, P., Nath Singh, Y., Gollapalli, R.P., Singh, S.P. (eds) Advances in Signal Processing and Communication Engineering. ICASPACE 2023. Lecture Notes in Electrical Engineering, vol 1157. Springer, Singapore. https://doi.org/10.1007/978-981-97-0562-7_46.

[17]. R. S. Kumar, K. K. Srinivas, A. Peddi and P. A. H. Vardhini, "Artificial Intelligence based Human Attention Detection through Brain Computer Interface for Health Care Monitoring," 2021 IEEE International Conference on Biomedical Engineering, Computer and Information Technology for Health (BECITHCON), Dhaka, Bangladesh, 2021, pp. 42-45, doi: 10.1109/BECITHCON54710.2021.9893646.

[18]. Vytla, V., Ramakuri, S. K., Peddi, A., Srinivas, K. K., & Ragav, N. N. (2021, February). Mathematical models for predicting COVID-19 pandemic: a review. In Journal of Physics: Conference Series (Vol. 1797, No. 1, p. 012009). IOP Publishing.

[19]. S. K. Ramakuri, C. Chakraborty, S. Ghosh and B. Gupta, "Performance analysis of eye-state charecterization through single electrode EEG device for medical application," 2017 Global Wireless Summit (GWS), Cape Town, South Africa, 2017, pp. 1-6, doi:10.1109/GWS.2017.8300494.

[20]. Dash M, Londhe ND, Ghosh S, Raj R, Sonawane R. Psoriasis Lesion Detection Using Hybrid Seeker Optimization-based Image Clustering. Curr Med Imaging. 2021;17(11):1330-1339. doi: 10.2174/1573405617666210224112123. PMID: 33655842.

[21]. Daniel, G. V., Chandrasekaran, K., Meenakshi, V., & Paneer, P. (2023). Robust Graph Neural-Network-Based Encoder for Node and Edge Deep Anomaly Detection on Attributed Networks. Electronics, 12(6), 1501. https://doi.org/10.3390/electronics12061501.

[22]. Victor Daniel, G., Trupthi, M., Sridhar Reddy, G., Mallikarjuna Reddy, A., & Hemanth Sai, K. (2025). AI Model Optimization Techniques. Model Optimization Methods for Efficient and Edge AI: Federated Learning Architectures, Frameworks and Applications, 87-108.

[23]. Lakshmi, M.A., Victor Daniel, G., Srinivasa Rao, D. (2019). Initial Centroids for K-Means Using Nearest Neighbors and Feature Means. In: Wang, J., Reddy, G., Prasad, V., Reddy, V. (eds) Soft Computing and Signal Processing . Advances in Intelligent Systems and Computing, vol 900. Springer, Singapore. https://doi.org/10.1007/978-981-13-3600-3_3

