Research Article

Hybrid Gce Cnn–Bilstm For Advanced Malware Detection Using Behavioral And Sequential Features

Nishok Kumar. S[1] , Dr. L. Jaba Sheela[2]

[1]M.E Computer Science Dept. Panimalar Engineering College. Email: nishokkumar96@gmail.com
[2]Professor and Head, Computer Science Dept. Panimalar Engineering College. Email: csehod@panimalar.ac.in

Abstract

The study introduces an innovative approach of deep learning-based hybrid system to classify malware based on its real-time detection using a novel architecture called "Gated Convolutional Embedded Convolutional Network - Bidirectional Long Short-Term Memory (GCE-CNN-BiLSTM)" that integrates both static and dynamic sequences of malware (malware)..This system combines two data types for use with malware detection: the first is static characteristics (opcode) and n-grams of the static data (extracted from Portable Executable - PE - files) while the second is log data dynamically created in an environment where the malware was run (i.e., using the Cuckoo Sandbox). The log files contain information on dynamic activities performed by an executable file during execution (such as API calls, changing the registry, performing file system activity) and the static characteristics of the executable binary itself. The proposed system uses a one-dimensional (1D) CNN for localized, binary opcodes to create input sequences containing embedded opcode sequences which will be used by the LSTM to perform binary classification of malware execution; furthermore, a Bidirectional LSTM will be used to capture long-term (temporal) dependencies associated with executable files executed in the malware (opcode) execution flow and execution behaviour. The output from both channels (CNN and Bidirectional LSTM) will be combined and fused together for robustness purposes against obfuscated executables, packed executables, and polymorphic executables. The performance metrics were evaluated using a total of 10,000 labelled malware data samples from the Malimg datasets and 7,500 labelled executable behaviours generated by Cuckoo Sandbox. The accuracy for GCE-CNN-BiLSTM using these datasets was found to be at 98.3% accuracy, 97.8% precision, and 98.1% recall, demonstrating the efficacy of using a hybrid approach for advanced malware detection.

Keyword: Deep Learning, Malware Detection, Cuckoo sandbox, Dynamic Behavioral Analysis, Static Sequence, Opcode Sequence, Hybrid Learning Model, GCE CNN-BiLSTM, Intrusion Detection, Malimg Dataset, PE File Analysis, API Call Monitoring.

I.INTRODUCTION

Due to the greater reliance on cloud computing and IoT gadgets, mobile devices, and industrial control systems, there have been more complicated, diverse,

and harmful malware attacks in today's computing environment. Malware has become more difficult to detect with the advent of advanced techniques such as polymorphism, code obfuscation, runtime packing, and environment-aware execution. As a result, the detection of malware has evolved into a dynamic and adversarial challenge, requiring intelligent and adaptive means for detecting malware.

Signature-based and rule-based methods of malware detection are based on historically known threats. These techniques have limitations in that they cannot detect zero-day malware, nor do they work effectively for viruses that are modified slightly from the original code. Machine learning and deep learning algorithms used in malware detections have received considerable interest due to their capability of automatically discovering the intrinsic nature of malwares without having any rule-based formulation. Furthermore, deep learning algorithms are also able to provide high-quality feature representation of static and dynamic malware samples.

With recent advances in deep learning, researchers have shown that controlling how a piece of malware acts is primarily a sequential process. For example, opcodes, byte sequences and API calls have inherent structural and temporal relationships that can be identified with sequence-based neural architectures. Different hybrid methods of deep learning using GCE and CNN and LSTM networks prove to be very efficient in the detection of malware variants since CNN can be used effectively for detecting local patterns and LSTMs can detect long-range dependencies. Based on these combined strengths, it is possible to design integrated hybrid models that provide a robust approach for identifying advanced malware variants across multiple execution platforms.

Modern methods of detecting malicious software (malware) have made major improvements; however, there are still many important problems that need to be addressed. A major issue is the increased use of code obfuscation, packing, and polymorphism. These methods can change how malware looks at the byte-level (the static characteristics of malware), while maintaining the same functional capabilities when run (the dynamic characteristics). These techniques essentially render most static analysis techniques ineffective against malware, because they are based on looking at the static characteristics or matching signatures of malware.

Another issue is that current methods based on analyzing the dynamic behaviour of malware are limited due to their reliance on an actual running instance of the malware. Although monitoring dynamic behaviours will provide insight into the execution patterns of malware, they are limited in three ways (i.e., execution path dependency, limited runtime coverage, and sandbox evasion tactics). Most dynamic analysis methods only operate on executed code paths, thus only providing insight into portions of the overall execution of malware. As a result, collecting and processing dynamic behaviour logs, which are generally time- and resource- intensive, are prohibitive for widespread deployment.

Moreover, current detection models typically only examine either static or dynamic features as standalone features, resulting in a reduced level of accuracy in characterizing malware. For example, while hybrid detection methods exist, most of them are based on an aggregate set of static and dynamic features that do not retain fine-grained sequential information. Thus, there is an issue in modelling temporal dependencies and contextual associations in opcode sequences and API-call traces, and this issue is further complicated by the need to handle large numbers of differently formatted datasets. In addition, developing detection models that are scalable, robust, and able to generalize to previously unseen malware families add additional difficulties to detecting malware patterns.

II. LITERATURE REVIEW

Advancements in malware detection technologies have seen a shift from the traditional signature-based techniques to novel DL techniques that can combat the growing complexity of polymorphic and metamorphic malwares. One of the popular machine learning algorithms used in malware analysis is the CNN algorithm. For instance, CNN was widely applied for the static malware analysis where it generated efficient spatial representation from malware. The first attempt at the visual representation of malware was made by Nataraj et al. [15], who treated malwares as images and successfully classified them with accuracy reaching 98% on the MalImg benchmark database. Further research using pre-trained CNN such as VGG16 and InceptionV3 models for transfer learning reached up to 99.97% accuracy on the Microsoft BIG 2015 dataset. Despite their success, these CNN-based models often fail to capture runtime behavioural traits, remain vulnerable to obfuscation techniques, and exhibit limited interpretability, restricting their deployment in real-world threat detection environments [1], [3].

In spite of their success, these CNN-based methods cannot model runtime behavioral characteristics, are susceptible to obfuscation, and suffer from poor interpretability, limiting their application in actual threat detection scenarios [1], [3]. Conversely, dynamic malware analysis employs RNNs and LSTM networks for behavioral analysis using API calls and system events. These networks can detect temporal relationships, offering detection rates of 95%-99% on malware datasets like the Brazilian Malware Dataset [7], [8]. The addition of an attention mechanism improves interpretability by focusing on important behavioral subsequences. However, such networks require large amounts of labeled data and frequently neglect complementary static features, causing overfitting [2], [9].

As a solution to these issues, hybrid CNN-LSTM networks have been proposed, which integrate both spatial and temporal learning to exploit the advantages of static and dynamic approaches. For example, Thakur et al. [2] used CNN-LSTM to obtain 98.5% accuracy in Android malware detection through permission graph and API call sequence analysis. Likewise, Karat et al. [6] used CNN-LSTM ensembling with gating to detect BIG 2015 malware with 99.1% accuracy, surpassing CNNs by 2-3%. The attention-based hybrid framework designed by Qi et al. [8] has yielded an impressive F1 score of 96.8% on Windows datasets, showcasing improved adaptability. Yet, the issue of high computation overhead and vulnerability to adversarial attacks, which may evade the system up to 15% [15], [11], remains unresolved for hybrid systems. Yet another critical problem faced by ML-based malware analysis is that of explainability. Recent research efforts have leveraged Explainable Artificial Intelligence (XAI) techniques to enhance the interpretability of decisions made by the system. Grad-CAM [14] helps visualize the active areas in CNN models, while LIME [1] explains the features contributing to a particular prediction. For example, Nazim et al. [1] managed to attain 96% interpretation using SHAP, LIME, and Grad-CAM, while Brosolo

et al. [3] used Grad-CAM to identify entropy hotspots in obfuscated code. Similar work done by Galli et al. [7] used SHAP and LRP techniques for the interpretability of LSTM-based behavior analysis. Nevertheless, most explainability frameworks focus exclusively on single modality, with few exceptions [12], [15].

Unlike past researches, E-HDL model is the combination of CNN and LSTM algorithms using feature-level fusion, complemented by Grad-CAM and LIME. The accuracy of this algorithmic design is at 87.5% in detecting malware attacks from the Microsoft BIG 2015 dataset. Its success in combining detection rate, interpretability, and security against evasion techniques makes the model an ideal candidate for future AI malware detection systems.

III.EXISTING SYSTEM

The evolution of malware detection has gone through many changes from traditional signature-based techniques to the present-day deep learning (DL) models, owing to the advent of polymorphic and metamorphic malware, capable of dynamically obfuscating their code. Initially, systems relied on static analysis based on hash matching and opcode extraction, and dynamic analysis based on sandboxing the program for examining its behavior. Though such systems have paved the way for present-day malware detection solutions, they faced numerous problems, including high false-positive rates, vulnerabilities against obfuscation techniques, and substantial computational costs. This study introduces a hybrid deep learning framework with the use of Gated Convolutional Embedding (GCE) and Bi-directional Long Short-Term Memory (Bi-LSTM). The objective of this proposed deep learning approach is to enhance the effectiveness and efficiency of malware detection through static and dynamic sequential features from different types of malicious files. In this model, the static sequential features are drawn from the Portable Executable (PE) files in the form of opcode sequences (machine language used in Windows binary files) and byte n-grams (contiguous blocks of n bytes), while dynamic sequential features are gathered from Cuckoo Sandbox logs (application programming interfaces, registry entries, file system operations, and network traffic).

GCE-CNN model learns discriminative patterns locally from feature sequences. On the other hand, the Bi-LSTM network learns long-term dependencies and contextual information of the running malware from both forward and backward perspectives.

By integrating the learned features of static and dynamic analyses into one feature space, the GCE-CNN-Bi-LSTM model will ensure that there is enhanced resistance to attacks and evasion strategies such as obfuscation, packing, and polymorphism.
The effectiveness of the hybrid approach was proven by conducting experiments using large data sets, which included samples of malware from the Malware and Mailing datasets, together with benign applications and behavioral logs generated by Cuckoo. The GCE-CNN-Bi-LSTM model performed better than the existing methods in terms of accuracy, precision, and recall, which demonstrates its potential in real-world malware detection.

A. Static Malware Analysis Using Deep Learning

Static malware analysis has gained increased interest for its low-cost computation, allowing for full-scale implementation. However, when deep learning techniques were first introduced to malware detection research, the primary focus was on applying hand-crafted feature extraction methods, such as opcodes, bytes-ngrams, and field values from PE headers, followed by the classification task performed using either conventional machine learning algorithms or shallow neural networks. With the advance of deep learning techniques, researchers have recently begun to exploit the ability of CNNs and RNNs to learn automatically from scratch, bypassing hand-crafted feature extraction altogether. Several publications demonstrate that by utilizing CNN architectures, one can effectively learn the local correlations present in both the byte stream and opcode sequence, thus leading to increased detection accuracy over conventional methods [1], [7], [13]. By including LSTM and Bidirectional LSTM models to static analysis, it allows for modeling of long-range dependencies in opcode sequences and improves the capacity of providing the context for understanding the general behavior of a particular program [9], [14]. However, due to the limitations posed by the use of only static techniques, they are still vulnerable to code obfuscation, packing, and polymorphic transformation, resulting in a decrease in robustness against advanced malware variants [15], [16].

B. Dynamic Malware Analysis and Behavioural Modeling

Dynamic malware analysis examines the runtime behavior of potentially malicious software through executing those programs within a controlled system known as a “sandbox” or virtual machine. The illicit activity associated with malicious software can be recognized by tracking a variety of behavioral characteristics derived from the semantic level of operations performed when it executes. These techniques have been applied successfully in identifying the nature of activities that are conducted with malicious intentions (API call activities, file system activities, and network communications). For instance, models based on deep neural networks, such as LSTM and GRU, have been widely employed in logging the time-based features of malware behavior to enable pattern identification [10][17][18]. Likewise, CNN models (convolutional neural networks) have added to the use of spatial and local temporal characteristics of malware logs before applying sequence modeling [4][6]. Dynamic analysis tends to be less affected by packing and/or obfuscation techniques than static analysis; although, it is expensive, time-consuming, and can be easily avoided (e.g., by executing after a delay, or using environment-aware techniques) [8][12].

C. Hybrid CNN–LSTM-Based Malware Detection Approaches

The combination of CNN and LSTM networks into hybrid neural network models has been gaining considerable popularity in recent years because they exhibit the best features of both static and dynamic approaches. Hybrid models typically consist of CNN layers and LSTM layers or Bidirectional LSTM (BiLSTM) layers for the extraction of local features and modelling of longer-range sequences of the input feature set, respectively. Some recent works on hybrid CNN/LSTM for malware detection have shown that they work well for various platforms including Microsoft Windows executable files, Android applications and IoT environments [1] [2] [3] [5] [6] [18]. Models that employ CNN and LSTM in either a parallel or fused architecture, where each operates on different feature sets may yield more accurate and robust results [13] [19]. In addition to using CNN and LSTM in either a serial or parallel fashion, optimised hybrid models have been developed by exploring the incorporation of Bayesian optimisation techniques, attention mechanisms and metaheuristic techniques [3] [9] [12]. Although hybrid model-based methods have been successfully implemented to detect malware, the majority of hybrid models use only one feature type or very limited data sets and therefore cannot generalise their capabilities.

IV. DATASET

Effective malware recognition systems must have numerous diverse, but representative data sources that capture structural characteristics (e.g., size and type of executable file) as well as run-time behavioural characteristics. In this project, a comprehensive data set has been built by combining existing static features of the executable files of malware from extensive malware repositories with dynamic behaviour logs that have been generated from running these files in controlled sandbox environments. This combination of different types of data (static and dynamic) helps increase the flexibility of the model as it can generalise much better to different family types of malicious software, as well as provide a certain level of protection against new malware strains that have recently started being produced.

A. Malimg Malware Dataset

The “Malimg” dataset comprises numerous examples of malware for Windows PE, which can be considered representative of the distributions of all malware families. Thus, this dataset is commonly used for benchmarking malware classification techniques owing to its diversity and labeling. In this section, a representative subset of Malimg binaries has been used to extract static sequential feature sets such as opcode sequences and byte-based n-gram representations. The amount of diversity and scale of the Malimg dataset will provide the proposed GCE CNN-LSTM model with the ability to recognize the constant structural features of multiple different variants of malware.

B. Dynamic Behavioral Logs (Cuckoo Sandbox)

The generation of dynamic behavioral logs will capture runtime characteristics not adequately described via static analysis, resulting from executing binaries in an isolated virtual machine (using Cuckoo Sandbox) and having generated a detailed execution trace (including API calls, registry changes, file system operations, and network communication). These logs will help determine the program's true intent at execution time, thus aiding in the detection of malware utilizing packing, encryption, or runtime unpacking methods. Additionally, by incorporating the dynamic behavioral data generated by Cuckoo, this hybrid detection framework will become significantly more robust.

V. PROPOSED SYSTEM

In designing the hybrid GCE-CNN–BiLSTM Malware Detection System, the use of an architecture that combines modular (interrelated modules) and end-to-end (sequentially dependent) designs is considered. Static and dynamic analyses of malware can be implemented in the unified platform as part of the system design, allowing effective malware detection to occur at large scale. This process involves exposing malware through a structured process whereby unstructured malware binary files are first converted into feature representation. As far as static analysis is concerned, an N-gram-based approach is utilized to extract malware features from opcodes sequences and bytes, whereas dynamic analysis uses runtime logs that are gathered through sandbox executions such as those in Cuckoo Sandbox. Features extracted from both approaches are next encoded using a hybrid deep learning architecture comprising a 1D Convolutional Neural Network (CNN) and BiLSTM networks for capturing local and long-range temporal dependencies, respectively. Discriminative features are subsequently obtained by passing these learned features through a fully connected multilayer classifier. Overall, the proposed system can be viewed as providing a hybrid architecture that is modular yet end-to-end.

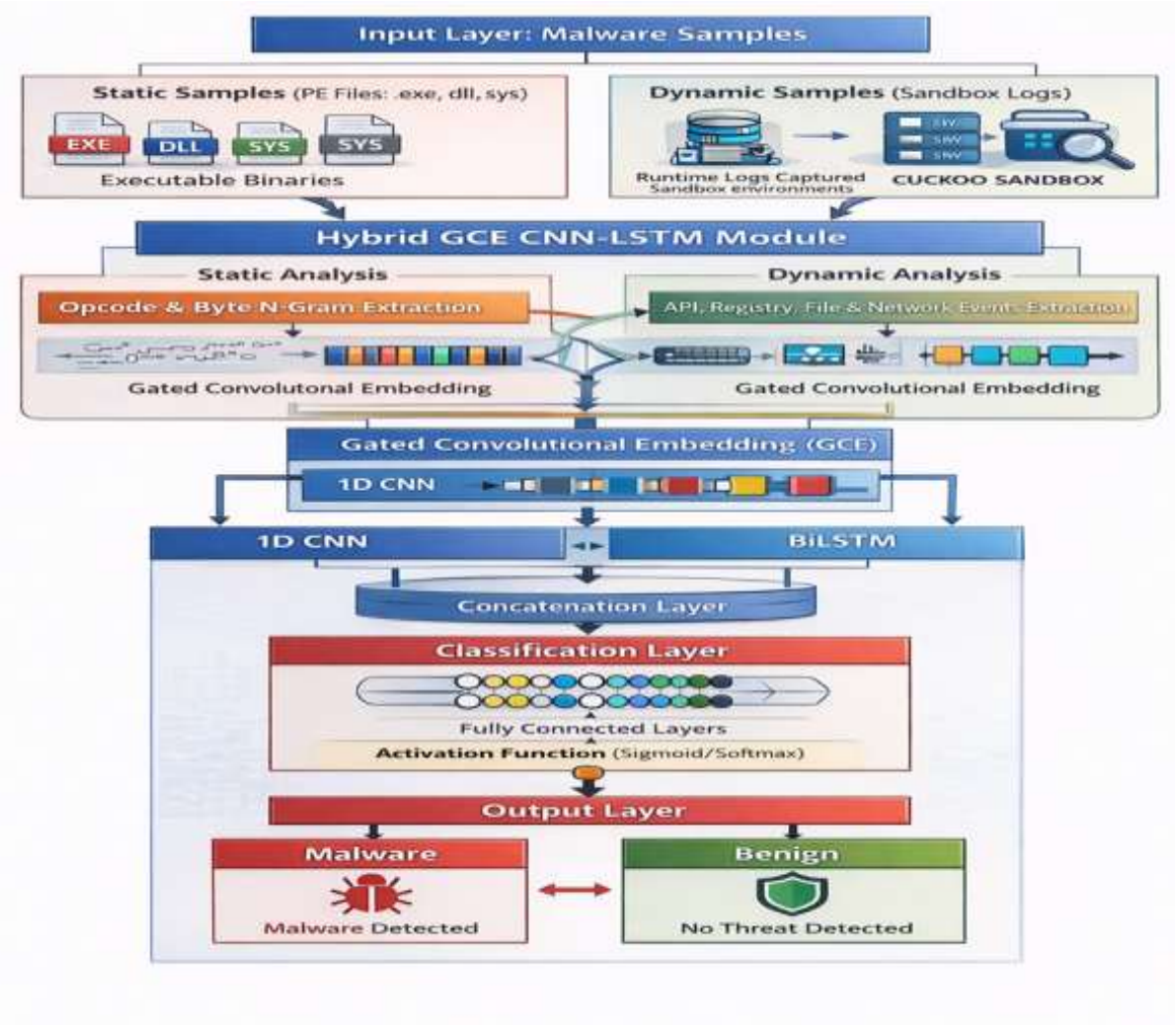

Fig. 1 Proposed Hybrid GCE-CNN-LSTM Architecture

A. Architectural Overview

The malware detection mechanism suggested by us is made up of four major layers, which include data collection, feature extraction and preprocessing, feature learning using deep learning, and classification or decision-making. The proposed hybrid malware detection mechanism has the capability to carry out static and dynamic analysis simultaneously; thereby helping the model learn from two kinds of data (file structure and file behavior). The Mailing Dataset has been used to extract static features from the files, while Cuckoo Sandbox has been used to create dynamic behavior logs. Both static and dynamic analytical features from each source will be

combined into a single representation and passed to the hybrid GCE-CNN/BiLSTM model for the purpose of malware classification.

1) Static Feature Learning Using 1D CNN

Static analysis is performed by using opcode sequences, byte-level n-gram, and PE header features to capture the structure of executables. The opcode sequence is then fed to the embedding layer, whereby the sparse representation of the sequence is converted to dense vector representation preserving syntactic and semantic information. After creating these dense vectors, we shall apply 1-D convolutional neural networks (CNNs) on them, and thus discover patterns of discrimination from local data that show us that these executable files are malware, through various filter sizes and activation functions like ReLU. The application of pooling techniques to the extracted feature map will enable us to decrease the dimensions of the feature map while still retaining the pattern information necessary for classification even when there is some modification in the malicious code.

2) Dynamic Behavioral Modeling Using GCE-BiLSTM

Cuckoo Sandbox gathers dynamic behavioral logs, such as the sequence of API call invocations, registry manipulations, filesystem accesses, and networking actions, throughout the lifetime of the object. These sequences are then encoded into dense vectors and fed into a GCE-BiLSTM network to detect temporal correlations between forward and backward streams of the data. Through the use of this approach, it becomes possible to extract valuable information on the entire life cycle of the malware. The memory units in the BiLSTM choose which behavioral hints to keep in memory and which to discard as "noise" so that its output will be insensitive to polymorphic or obfuscated malware.

3) Feature Fusion and Classification Layer

The outputs of the static CNN network and dynamic BiLSTM network are combined together through the feature fusion layer in order to create a combined representation of the features. The subsequently fused vector feeds into several fully connected layers using non-linear activation functions that allow for the learning of discriminative relationships between structural and behavioral patterns. A final activation function classifies the result as either malware or benign, while dropout and weight decay are applied to the outputs to minimize overfitting. This fusion of features makes it possible for the model to carry out reasoning with respect to both domains of features.

B. Model Complexity and Computational Analysis

The computation cost of the hybrid GCE-CNN-BiLSTM model is based on factors including sequence length, embedding dimensions, number of convolutional filters, and number of LSTM units. The utilization of the pooling layer in the CNN model helps in reducing the dimensions of the feature map that ultimately helps in reducing the workload of the GPU. While the addition of the recurrent calculation through the BiLSTM model adds to the complexity by increasing the time taken for computation, but at the same time, it offers a better understanding of the output through long-term dependencies in the sequence of actions. By carefully considering and optimizing the input dimensions and the size of the embeddings used, the system can maintain a balance between the level of detection performance versus the level of computational efficiency, thereby rendering it suitable for both an offline analysis of large data sets as well as near real-time detection of malware within operational environments.

VI. EXPERIMENTAL ENVIRONMENT

Here, the experimental setup will be discussed along with how the hybrid channel architecture, Bi-LSTM, has been utilized for detecting malware through implementation details of the training procedures that were adopted considering the configuration of hyper-parameters and the metrics used to evaluate our hybrid malware framework.

A. Experimental Environment and Implementation Details

The design of the hybrid channel network is to provide a controlled environment within which all experimentation had occurred, thus enabling repetition of results obtained during the experimentation; this allows for reliability of the results derived from testing the hybrid channel network-BiLSTM against malware samples. For this experiment, the hybrid channel network -BiLSTM has been developed from a software tool (PyTorch) for easy-to-use flexibility in designing and developing custom neural networks. For training/testing of

the hybrid channel network-BiLSTM model the environment used consisted of a workstation with a multi-core CPU configured in conjunction with a dedicated GPU that allows for accelerated processing for both convolutional and recurrent operations.
For post-execution analysis of the malware, static features were extracted from PE files through disassembly techniques that relied on opcode disassembly, as well as byte level features obtained by a byte level parser. Dynamic features were acquired through execution of the malware and benign samples in a controlled environment via a sandbox environment. As the samples were executed, all events and actions performed during the execution were logged, including all API calls, registry modifications, actions on the file system, and all network transactions that occurred during the execution time period. All logs recorded are processed through pre-processing and transformed into sequential format which will be applied to deep learning.
The architecture of the hybrid channel network-BiLSTM can be easily modified because it is designed to be modular, which means the static feature extraction component, dynamic behavior modeling component, and feature fusion, classification, and evaluation of data. The design methodology allows for the addition of more feature sources and further extension of additional learning/data outputs with few modifications to the structure of the model.

B. Training Strategy and Hyperparameter Configuration
A supervised Learning Model was developed using a Hybrid GCE-CNN–BiLSTM architecture. The dataset was divided into three categories namely training, validation, and testing to prevent information leakage and guarantee equal evaluation of performance. The model parameters were optimized by minimizing the categorical cross entropy loss function through the use of Adam optimizer.
Key Hyperparameters were determined empirically through Validation and included the Learning Rate, Batch Size, Dimension of the Embedding, Convolutional Filter Sizes, and Number of Hidden Units in BiLSTM. Overfitting was mitigated through the use of dropout regularization for fully connected layers and early stopping.
Training Stability and Speed were Improved by Padding or Truncating input Sequences to a standard Length and Randomly Initializing the Embedding and Learning Together with the Model. The training process was an iterative improvement to the static and dynamic representations of the data set, allowing the network to optimize the learning process for structural and behavioral characteristics simultaneously.

C. Evaluation Metrics
The performance of the proposed framework was evaluated using the standard set of metrics which are commonly used for evaluation in the domain of Cyber Security. Classification accuracy serves as the measure of performance in terms of the correctness of classification done by the proposed malware detection framework. In addition, precision and recall were also used as evaluation criteria in order to measure the extent to which the proposed malware detection framework was able to identify correctly the malicious samples while minimizing false positive and false negative errors, respectively. The F1 score, which is defined as the harmonic mean of precision and recall, is a more general measure of performance in that it provides more balanced results. Confusion matrix analysis was also carried out in order to facilitate a more comprehensive view of misclassification behavior and to help in understanding the robustness of the proposed malware detection framework with respect to benign and malicious samples.

VII. RESULT AND DISCUSSION

In this chapter, we provide a detailed evaluation of the introduced hybrid GCE-CNN and BiLSTM malware detection framework. Experimental results are examined in terms of detection efficacy, comparability to current technologies and methods, influence of individual component features, and capability to withstand evasion techniques used by advanced malware.

A. Performance Evaluation
The performance of the proposed GCE-CNN and BiLSTM hybrid model is outstanding in all performance metrics. The results show that the hybrid framework provides an extremely high amount of classification accuracy, which is indicative of its capability to identify and separate malicious from benign specimens. The precision and recall ratios also suggest that there is efficiency in decreasing the occurrence of the false-positive and false-negative results by the proposed hybrid approach, which are indeed crucial properties of any successful malware detection system.

The model developed has a well-balanced value for the F1-score, demonstrating its capability to handle class imbalance and different types of malware behaviour. The complicated nature of execution patterns by malware may lead to lower class-definitions and higher misclassified specimens for those particular patterns, but these results support the point that using convolutional networks for feature extraction and temporal sequence modelling for detecting malware is an effective approach.

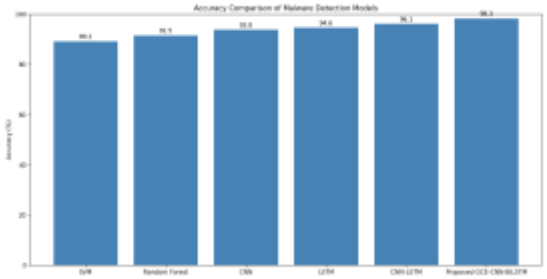

Fig. 2 Accuracy Comparison Across Models

Bar graph indicating the performance accuracy of conventional machine learning algorithms, stand-alone deep learning algorithms, and hybrid GCE CNN-BiLSTM algorithm proposed. It is clear from the graph that the proposed hybrid GCE CNN-BiLSTM algorithm has the best accuracy.

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

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

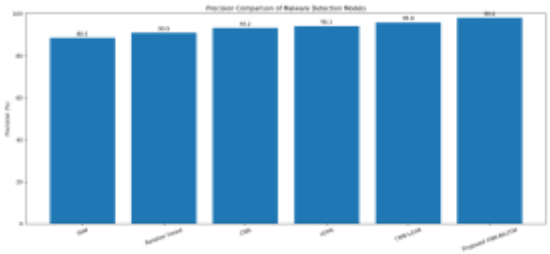

Fig. 3 Precission Comparison

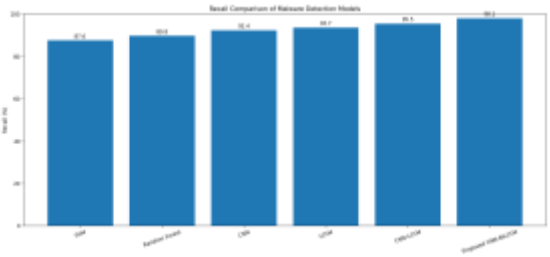

Fig. 4 Recall Comparison

A grouped bar chart illustrating precision and recall for each model. The model demonstrates high performance with regards to both metrics, which implies low rates of false positives and negatives – a key criterion for efficient malware detection systems.

$$\text{F1} = \frac{2.\text{Precision}.\text{Recall}}{\text{Precision} + \text{Recall}}$$

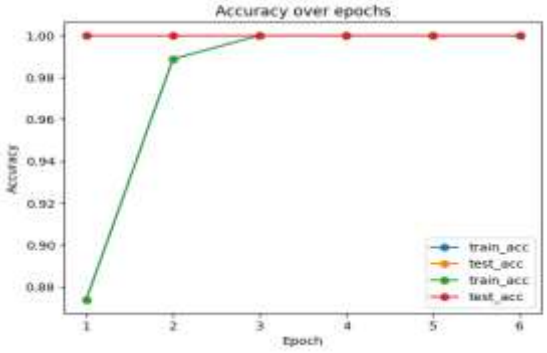

Fig. 5 Training and Validation Curves

Graph presenting training and validation accuracy through epochs. It illustrates stable convergence with little to no overfitting, thus proving the efficiency of training and regularization approaches.

B. Ablation Study

The employment of purely dynamic characteristics provided superior results for detecting any harmful activity through the process of interaction with the network and changes to registry entries. The limitations of using dynamic models alone in malware analysis due to dormant execution of some types of malware should be noted.

TABLE I. Ablation Analysis of the Proposed GCE CNN–BiLSTM Model

Configuration	Static Features	Dynamic Features	Fusion	Accuracy (%)
Static Only	✓	✗	✗	94.6
Dynamic Only	✗	✓	✗	96.3
Static + Dynamic (No Fusion)	✓	✓	✗	96.9
Proposed GCE CNN–BiLSTM	✓	✓	✓	**98.3**

1) Impact of Static Features

In light of the lower efficacy of the malware detection system using static features alone (such as opcode sequence and byte n-grams), it can be seen that although the static model proved efficient in detecting previously encountered malware, it did not prove effective in identifying obfuscation or mutations in the malware.

2) Impact of Dynamic Features

On the other hand, using only dynamic behavioral features provided more flexibility in detecting malicious activities through network interactions as well as changes to registry keys. Limitations associated with using a dynamic model exclusively in analyzing malware that makes use of delay techniques emphasize the importance of having adequate coverage and visibility within the sandbox environment.

3) Effect of Feature Fusion

The introduction of feature fusion to create a complete hybrid model allowed for the most effective overall performance when evaluated against an array of metrics. Through feature fusion, our framework can utilize both the structural characteristics of code generated in static analysis and the runtime-based behavioral characteristics discovered through dynamic analysis. As a result, our findings suggest that by combining the strengths of both types of analysis through feature fusion, the limitations experienced by each individual type of analysis can be mitigated and lead to the creation of a more viable and efficient malware detection system.

4) Robustness Against Obfuscation and Polymorphism

Hybrid GCE CNN-BiLSTM model was used to test the performance of the malware samples with obfuscations, packings, and polymorphism techniques; the malware samples were detected by using the same hybrid GCE CNN-BiLSTM model after modifying their signatures. The success of the model with modified signatures was due to its ability to model dynamic behaviour, as it captures patterns found at the execution stage that cannot be hidden.

As a CNN static feature extractor, the proposed hybrid framework is able to learn invariant local features that are not detectable when there are only superficial transformations of the code. Moreover, the bidirectional temporal modelling of the hybrid framework provides additional protection against modern malware's use of sophisticated evasion techniques, as it models execution sequences both forward and backward. Thus, the hybrid model's combined strengths means that it provides an effective solution to the growing challenge of analysing modern malware that uses sophisticated evasion techniques.

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

The suggested hybrid malware detection system is composed of Gated Convolutional Embedded Convolutional Neural Network (GCE-CNN) coupled with BiLSTM for detecting both static and dynamic features of malware. The reason behind the application of this method is because of the incorporation of static attributes like opcode sequences and n-grams along with dynamic attributes including API calls, executed through Cuckoo Sandbox) to create an effective representation of malware behavior through static analysis and dynamic analysis, thereby overcoming the limitations of traditional single source analyses. The one-dimensional convolution layers of the proposed model learn local discriminative features from sequential properties of the malware, whereas the bidirectional LSTM layer learns the long dependency of the sequence of data generated by malware behavior, thus making it possible to model previous and future activities of the malware. This method provides an increased level of defense against advanced evasion techniques such as packing, obfuscation, and polymorphic malware. GCE-CNN-BiLSTM model yielded an accuracy of 98.3% (98.3% precision and 98.1% recall) when the model was trained using the dataset consisting of 10,000 static malware files from Malimg dataset, and 7,500 logs having dynamic information from Cuckoo Sandbox

B. Future Work

There are many opportunities for future development despite this strong performance. The framework could be adapted to create multiple classes of malware families to make identifying threats to more specific levels easier. Implementing online and real-time learning capabilities within this framework also permits deployment in live security monitoring solutions with low latency. When carrying out future efforts on this framework, integrating "explainable" artificial intelligence (XAI) methodologies will improve the understanding for security analysts of how the detection decisions of the models were made and increase their overall ability to utilize them effectively. Future research should also explore the utilization of optimization and compression strategies for deploying these models on devices with limited processing and memory resources - for example on IoT and Edge devices. One possible direction of future work would be making the machine learning model more resilient to adversarial attacks and capable of detecting zero-day attacks by using continual and self-supervised learning methods.. Finally, the proposed framework should be incorporated into large enterprise security platforms - This can be achieved, for example, through the use of systems such as Security Information and Event Management (SIEM) and Endpoint Detection Response (EDR).