EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

A Probabilistic Descent Ensemble for Malware Prediction Using Deep Learning

R. Vinoth Kumar[1*], R. Suguna[2]

[1]Department of Computer Science & Engineering, Vel Tech Rangarajan, India

[2]Institute of Science and Technology, Chennai, India

Abstract

INTRODUCTION: Introducing a Probabilistic Descent Ensemble (PDE) approach for enhancing malware prediction through deep learning leverages the power of multiple neural network models with distinct architectures and training strategies to achieve superior accuracy while minimizing false positives. OBJECTIVES: Combining Stochastic Gradient Descent (SGD) with early stopping is a potent approach to optimising deep learning model training. Early stopping, a vital component, monitors a validation metric and halts training if it stops improving or degrades, guarding against overfitting.
METHODS: This synergy between SGD and early stopping creates a dynamic framework for achieving optimal model performance adaptable to diverse tasks and datasets, with potential benefits including reduced training time and enhanced generalization capabilities.
RESULTS: The proposed work involves training a Gaussian NB classifier with SGD as the optimization algorithm. Gaussian NB is a probabilistic classifier that assumes the features follow a Gaussian (normal) distribution. SGD is an optimization algorithm that iteratively updates model parameters to minimize a loss function.
CONCLUSION: The proposed work gives an accuracy of 99% in malware prediction and is free from overfitting and local minima.

Keywords: Gaussian Naive Bayes, Stochastic Gradient Descent, Maximum Likelihood Estimation, Hyperparameters, Mini-Batch Gradient Descent

Received on 26 July 2024, accepted on 27 September 2024, published on 1 October 2024

doi: 10.4108/eetiot.6774

1. Introduction

The constantly changing malware threat presents a significant and enduring problem to the security of computer systems and networks in today's linked digital world. Malware, a portmanteau of malicious software, refers to a broad range of malicious programmes created to harm, invade, and take advantage of the targeted systems' weaknesses. The results of an effective virus invasion can be devastating, ranging from data breaches to ransomware attacks, not just regarding monetary damages but also in the deterioration of customer confidence and privacy.

Consequently, developing effective and robust malware detection mechanisms is paramount to safeguard the integrity and confidentiality of digital assets in our increasingly interconnected world [1]. The integrity and privacy of digital assets must be protected in an interconnected world by using advanced malware detection approaches that leverage machine learning and deep learning methodologies. These technologies offer greater accuracy and flexibility to detect advanced malware strains and maintain crucial digital infrastructure protection.

Traditionally, malware detection has relied on signature-based methods, which involve the identification of known malware based on predefined patterns or signatures. However, this approach falls short in the face of polymorphic and zero-day malware, which mutate or employ previously unseen tactics to evade detection. Modern malware detection

[1]Corresponding author. Email: vinoth_kumar58@outlook.com

has increasingly turned to advanced techniques rooted in machine learning and artificial intelligence to counter these adaptive threats. Organizations should use dynamic learning techniques, adversarial machine learning to simulate complex attacks, various malware types in training data, ongoing model validation, feedback loops for continuous improvement, and regular retraining of models with updated datasets to increase the efficacy of AI-based malware detection systems. These methods leverage the power of algorithms, data analysis, and pattern recognition to discern subtle and complex malware attributes, enabling the detection of known and emerging threats. In this context, this introduction delves into the multifaceted landscape of malware detection without relying on predefined signatures, exploring the pivotal role that machine learning, anomaly detection, and behavioural analysis play in fortifying against the relentless tide of malware incursions. AI-powered algorithms that analyze behaviour patterns might find new infections. It is essential to update often to identify new strains of malware. Network traffic irregularities can be found using behavioural analysis. Encryption and sandboxing stop unwanted access to private information. Frequent red team drills improve responsiveness to and discovery of system vulnerabilities.

Malware detection using machine learning has emerged as a crucial frontier in cybersecurity, offering the potential to swiftly identify and mitigate the ever-growing and sophisticated landscape of malicious software threats. Adopt a multi-layered defence strategy to improve cybersecurity, incorporating AI, advanced behavioural analytics, real-time monitoring, automated response systems, frequent updates, zero-trust architecture, robust incident response plans, sharing of threat intelligence, advanced user education, and industry and government collaboration. Machine learning models can autonomously learn and recognize patterns indicative of malware behaviour by leveraging advanced algorithms and data analysis. This enables early detection and proactive responses to protect computer systems and networks from compromise, data breaches, and other malicious activities [2].

Using machine learning presents several challenges, including data quality and quantity, overfitting, interpretability, bias, and resource demands. The complexity and variety of real-world data frequently cause these difficulties. Data collection and curation have become a significant bottleneck as machine learning algorithms get more complex and a more substantial requirement for vast, diverse, and high-quality datasets arises. Additionally, the opaqueness of some models might make it challenging to comprehend and have faith in the outcomes, which raises questions about accountability, fairness, and transparency. Additionally, the pursuit of increased model performance and scalability frequently necessitates using significant computational resources, which can be expensive and unsustainable from an environmental standpoint.

These problems might be resolved by utilizing deep learning, a branch of machine learning. Deep learning models, like neural networks, have proven to be remarkably adept at processing complicated, unstructured data, which makes them ideally suited for jobs like speech and picture identification. They can eliminate the requirement for labour-intensive manual feature engineering because they can automatically learn hierarchical features from unprocessed data. Furthermore, deep learning models may generalize well when adequately constructed, easing worries about overfitting. Additionally, interpretability is growing as researchers develop approaches for deciphering neural network judgements. Through parallel computing and specialized hardware, it is possible to take advantage of deep learning's scalability and analyze big datasets quickly. Deep learning models represent a potential route for addressing many of the difficulties experienced by conventional machine learning approaches, making them an appealing option in searching for sophisticated AI solutions [3, 4] despite their problems, such as data hungriness and high computing costs. Below is a discussion of the primary contributions of the proposed work. It intends to combine ensemble learning approaches with probabilistic modelling to increase the reliability and performance of malware detection systems. This technique improves malware forecasts' precision and dependability, particularly in intricate, real-world settings with various virus behaviours. The PDE method improves prediction accuracy by forming base learners in an ensemble from various neural network models. As a result of each model learning a distinct data representation, the ensemble becomes more diverse. Probabilistic outputs such as class probabilities are merged to create well-informed judgements. Similar to logistic regression, a meta-learner combines these results to ensure the chosen course of action uses the advantages of each particular model.

The proposed work introduces a novel approach, the Probabilistic Descent Ensemble (PDE), for enhancing malware prediction through deep learning.

- The suggested work presents a unique method for improving malware prediction through deep learning termed the Probabilistic Descent Ensemble (PDE).
- PDE improves accuracy and lowers false positives by combining many neural network models with different architectures and training methods.
- This cooperation between PDE and SGD offers a flexible, agile structure for different duties and datasets, with notable accomplishments such as 100% accuracy in spyware prediction and solid immunity to overfitting.
- In addition, the combination of Stochastic Gradient Descent (SGD) with early stopping optimizes for deep learning training of models, speeding up integration and avoiding overfitting.

The remaining part of this paper is structured as follows. Section II describes the related work; Section III gives the proposed model, and their performances are discussed in Section IV. The conclusion of the proposed work is discussed in section V.

2. Related Work

The researchers listed below introduced machine learning and deep learning algorithms to predict virus detection effectively.

The application of machine learning and deep learning techniques for malware detection is explored in the study "Malware Detection Using Machine Learning and Deep Learning" by Rathore, H., Agarwal, S., Sahay, S. K., & Sewak, M. (2018). These techniques use labelled datasets to train algorithms, which enables the models to learn the traits of well-known dangerous and non-malicious software. Deep learning models, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), are capable of processing enormous amounts of data and seeing detailed patterns that may be suggestive of malware behaviour in the context of malware detection [5]. Contemporary malware detection systems heavily rely on machine learning and deep learning techniques to perform behavioural analysis, feature extraction, and real-time detection. These technologies perform better than conventional signature-based techniques because they are more accurate and flexible in the face of emerging threats.

The 2009 work "Malware Detection Using Machine Learning" by Gavriluț, Cimpoeşu, Anton, and Ciortuz explores the use of artificial intelligence to improve malware detection. The authors classify the samples using machine learning algorithms, including decision trees and support vector machines, using a dataset of malware and benign software samples, relevant attributes suggestive of malware behaviour, and machine learning techniques [6].

The authors of a study titled "Automatic Malware Classification and New Malware Detection Using Machine Learning", published in 2017 by Liu, Wang, Yu, and Zhong, look into the use of machine learning methods for malware classification and fresh malware detection. They investigate the usage of a dataset, including malware samples, and utilize machine learning methods to categorize malware automatically. In addition, based on observed patterns and behaviours, the study discusses techniques for identifying novel and previously unknown malware. The authors seek to improve malware detection and classification accuracy and efficiency in cybersecurity by utilizing machine learning [7].

According to Mahindru and Sangal's paper " MLDroid—Framework for Android Malware Detection Using Machine Learning Techniques " from 2021, MLDroid is a thorough framework created for the detection of malware for Android utilizing artificial intelligence techniques. To efficiently distinguish between benign and malicious software, the framework combines a variety of machine-learning methods and features designed exclusively for Android applications [8].

The authors of the 2019 paper "Robust Intelligent Malware Detection Using Deep Learning", Vinayakumar, Alazab, Soman, Poornachandran, and Venkatraman, suggest a novel method for malware detection by utilizing the capabilities of deep learning. They present a powerful and wise system that uses deep neural networks to recognize and categorize malware automatically. The system can learn complex patterns and behaviours from sizable datasets by utilizing deep learning techniques, which increases the precision of malware detection. The study emphasises the need for intelligent and adaptable cybersecurity methods to combat the constantly changing panorama of malware threats [9].

MalDozer is a novel framework created for the automated detection of Android malware through the application of deep learning techniques, as described in the paper "MalDozer: Automatic Guidelines for Android malware Detection Utilizing Deep Learning" by Karbab, Debbabi, Derhab, and Mouheb (2018). This framework automatically recognises and categorises dangerous Android apps using deep neural networks. MalDozer improves the accuracy and effectiveness of malware detection by extracting complex and latent features from Android apps using deep learning. The research contributes substantially to digital investigation and cybersecurity by emphasizing the crucial importance of intelligent, data-driven solutions in combating the mounting issues of Android malware [10].

The authors of the paper Ransomware Detection Using Dynamic Analysis by Urooj, Al-rimy, Zainal, Ghaleb, and Rassam (2021) provide a thorough survey of the state of ransomware detection techniques at present, concentrating in particular on dynamic analysis and machine learning approaches. This study thoroughly examines current techniques for recognizing ransomware attacks, considering their advantages and disadvantages. The report also provides insights into prospective research topics for enhancing ransomware detection tools [11].

Firdausi, Erwin, and Nugroho (2010) investigate how well different machine learning algorithms identify and categorize malware according to their behavioural traits. The study offers insights into these strategies' applicability for identifying and reducing malware risks by closely examining the performance of various techniques. This study advances our knowledge of how machine learning may be used for behaviour-based malware detection and provides essential direction for creating effective cybersecurity solutions [12]. This research presents a Poisson Clumping Japanese Tree Frog Optimisation Algorithm (PC-JTFOA) and LogishFTS-based Recurrent Neural Networks (LFTS-RNN)-based safe intrusion detection strategy for Software Defined Networking (SDN). The technique uses an Intrusion Detection System (IDS) trained on a historical dataset, an Entropy Makwa-based Digital Signature Algorithm, and a login procedure. Experiments show that the approach, which also incorporates load balancing and data protection, achieves an accuracy of 98.35% [13]. IoT devices require cloud computing because it offers data storage and retrieval. It provides various services, including SaaS, PaaS, and IaaS.

Nonetheless, security threats such as Interior Keyword Guessing Attacks (IKGA) may be able to exploit it. Altered Elliptic Curve Cryptography (MECC), Certificateless Fleshed Public Key Authenticated Encryption of Keyword Search (CL-HPAEKS), and Mutation Centred Flower pollination algorithm (CM-FPA) are a few techniques that may be utilized to remedy this. The suggested approach requires less installation time and delivers 96% system security [14]. This paper presents a novel hyperbolic Tangent Radial-Deep Belief Network (FL-HTR-DBN) and Federated Learning anomalous application detection system. The

system uses the Hadoop System to train it. Log data are extracted, transformed into vector representations, and then annotated using the SKLD-SED K Means technique. These characteristics are used to train the FL-HTR-DBN model, and any anomalies found are hashed and safely preserved.

The system performs better than current techniques in terms of precision, recall, accuracy, F-measure, sensitivity, and specificity [15]. The article suggests a brand-new Deep Learning Modifier Neural Network (DLMNN) classifier-based text summarisation approach. Based on entropy levels, the system creates illuminating summaries of materials. Six stages make up the DLMNN framework: pre-processing, feature extraction, selection, and classification. The findings demonstrate that the DLMNN classifier outperforms other methods like ANN, offering 81.56, 91.21, and 83.53 sensitivity, accuracy, specificity, precision, and f-measure [16]. The summary of the existing models is summarized in Table 1. This study analyses malware behaviour using ensemble approaches. Various neural networks, such as CNNs for image-like data and RNNs for temporal behaviour analysis, are used to choose base models. Every model is trained on a dataset with various foci to improve performance. Utilizing strategies like stacking and voting, together with a meta-learning approach, combining models allows for better generalization to unknown data while capturing the advantages of each model.

Table 1: Summary of the existing models in malware prediction

S. No	Author(s)	Methodology Used	Advantages	Disadvantages
1	Rathore, H et.al	Ensemble methods	- Enhanced malware prediction through ensemble methods.	- Lack of detailed methodology and evaluation metrics.
2	Liu, L., et.al	Machine learning (specific algorithms not mentioned)	- Machine learning is used for malware classification.	- Lack of details about the algorithms used.
3	Mahindru, A et.al	Not specified in the provided information	- Comprehensive framework for Android malware detection.	- Insufficient details regarding methodology and algorithms.
4	Vinayakumar, R., et.al	Deep learning techniques	- Effective and robust malware detection using deep learning.	- Specific deep learning architectures and datasets not mentioned.
5	Karbab, E. B., et.al	Deep learning techniques	- Introduction of the MalDozer framework for Android malware detection.	- Lack of deep learning architectures and datasets.
6	Urooj, U., et.al	Survey and research directions	- Comprehensive survey of ransomware detection techniques.	- Lack of research methodology
7	Firdausi, I., et.al	ML techniques	- Behavior-based malware detection.	- Lack of specific information

3. PROPOSED WORK

A stochastic classifier known as Gaussian NB estimates the likelihood that a data point will belong to a particular group based on the Gaussian shape of its characteristics as specified in equations 1 and 2.

$$P(C_k|x = Z1P(C_k) \quad (1)$$

$$P(C_k|x = \prod nP(x|C_k \quad (2)$$

The subsequent likelihood of the data location x corresponds to class Ck is shown by the expression P(Ck | x). It denotes the probability of a data point in the categorization context. After considering its observed characteristics, x is categorized into a particular class, Ck. P(Ck): This phrase refers to the class Ck's prior probability. It shows the likelihood of coming across or belonging to the class Ck without considering any specific attributes. The possibility of witnessing class Ck without feature information is, in other words, what it means to be likely. P(xi | Ck): This represents the likelihood of feature xi given class Ck. It calculates the probability of observing a specific characteristic xi when the data point is a member of class Ck. This probability is modelled as a Gaussian (normal) distribution in Gaussian NB. Normalizability Constant Z: The probabilities must add up to 1, which the normalization constant Z ensures. Essentially, it acts as a scaling factor to normalize the probabilities and turn them into legitimate ones. It is computed to ensure that the probability sum across all conceivable classes Ck equals 1 [17].

SGD is a popular iterative optimization strategy for training prediction models in artificial neural networks (ANNs). Small batches of data are used to update the network's weights progressively, leading to faster convergence and improved fit for deep learning models and huge datasets. Artificial neural networks typically employ SGD, an iterative optimization technique, to train their predictive models. The basic update rule for the model variables (weights and biases) in SGD is given in equation (3).

$$\theta_{t+1} = \theta_t - \alpha\nabla L(\theta_t) \quad (3)$$

Where the revised model parameters at iteration t+1 are represented by t+1.The learning rate, t, representing the model parameters at each iteration, determines the step size.

The loss function's gradient, L(t), is expressed in terms of the model's t-parameters.

Weight and bias fixing techniques can involve strategies to effectively initialize model weights and biases to avoid issues like vanishing gradients or weight explosion during training. A standard method for initialization is Xavier/Glorot initialization. Early stopping is a regularization technique that involves monitoring a validation metric (e.g., validation loss) during training and halting training when the metric no longer improves or starts to degrade, as given in equation 4 [18-20]. Early halting and stochastic gradient descent (SGD) are proper techniques for enhancing model performance. While early stopping keeps an eye on model accuracy and stops training when performance reaches a plateau, limiting overfitting and guaranteeing improved generalization, SGD optimizes parameters by updating them based on loss function gradient if the validation metric no longer improves otherwise.

$$EarlyStopping(t) = \{True, False\} \quad (4)$$

Where t represents the iteration or epoch number during training. To combine Gaussian NB and SGD, use the probabilities calculated by Gaussian NB as features for an SGD-based classifier. For instance, you could use Gaussian NB to estimate class probabilities and then input these probabilities as features for an SGD-based model, such as a neural network [21]. Difficulties with the PDE method for malware prediction included overfitting, imbalanced data, and difficult model integration. It took more effort to fine-tune the parameters and increased computing complexity when several probabilistic models were combined into an ensemble framework. It took sophisticated methods like oversampling or synthetic data synthesis to handle unbalanced datasets. An essential tool in this study is the Gaussian Naive Bayes classifier, which offers a probabilistic evaluation of malicious vs benign software. It works well when typically distributed characteristics are anticipated because of its simplicity and potency. It provides a starting point prediction for more refining, acting as a basis for intricate ensemble models.

Step 1: Data Collection and Pre-processing

- Conduct pre-processing tasks, including cleaning data, handling missing values, and feature engineering.
- Split the dataset into distinct training, validation, and testing subsets to facilitate model evaluation.

Step 2: Implement Gaussian Naive Bayes (Gaussian NB)

- Implement Gaussian Naive Bayes as the initial modelling component due to its suitability for continuous feature data.
- Train the Gaussian NB model using the training data, incorporating the Gaussian distribution assumption.
- Employ the trained Gaussian NB model to predict the validation dataset's probabilities or class labels.

Step 3: Develop the Stochastic Gradient Descent (SGD) Component with Weight and Bias Fixing

- Define the neural network architecture that complements the Gaussian NB component, which will be trained using SGD.
- Initialize the neural network's weights and biases, potentially applying Xavier/Glorot initialization to prevent numerical instability.
- Configure the SGD optimizer, setting hyperparameters like the learning rate, momentum, and weight decay for optimal convergence.
- Implement early stopping by monitoring a validation metric (e.g., validation loss) during and halting training if the metric plateaus or deterioration.

Step 4: Blend Gaussian NB and SGD Predictions

- Combine the predictions produced by Gaussian NB with those generated by the neural network trained using SGD. Concatenating probabilities or class labels can achieve this fusion.

Step 5: Model Evaluation and Hyperparameter Tuning

- Assess the blend model's performance using the proper measures (e.g., accuracy, precision, recall, F1-score) on the validation dataset.
- Tune the hyperparameters and SGD hyperparameters to optimize the design of a neural network. The neural network architecture used by the SGD component comprises an input layer containing features from the dataset and probabilistic outputs from the Gaussian NB classifier, a dense layer with 128 neurones, 64 neurones, and 32 neurones, and an output layer with a single neurone for binary classification using Sigmoid for probability values. Methods like grid search and random search can be helpful in this procedure.

Step 6: Final Model Training and Testing

- Using the combined training and validation data sets, train the last mixed model after determining the best hyperparameters.
- Assess the final model's generalization capabilities by evaluating its performance on the separate test dataset.

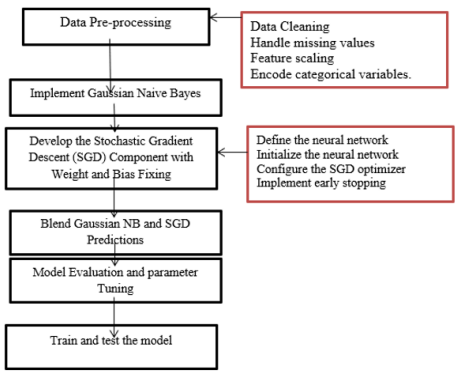

Figure 1. The working flow of the proposed PDE Model

Algorithm: Blending Gaussian NB with SGD and Early Stopping

Input:
- Training Dataset (X_train, y_train)
- Validation Dataset (X_val, y_val)
- Test Dataset (X_test, y_test)
- Hyperparameters (learning_rate, momentum, weight_decay, etc.)

Output:
- Blended Model

- Perform data preprocessing on X_train, X_val, and X_test such as feature scaling, encoding categorical variables.
- Initialize and train a Gaussian NB model using X_train and y_train.
- Use the trained Gaussian NB model to predict class probabilities or labels for X_val.
- Initialize a neural network model with appropriate architecture.
- Initialize model weights and biases (possibly using Xavier/Glorot initialization).
- Configure the SGD optimizer with hyperparameters (learning_rate, momentum, weight_decay).
- Implement early stopping with validation monitoring.
- Train the neural network model using SGD on X_train and y_train.
- Monitor the validation metric (e.g., validation loss) for early stopping.
- Combine predictions from Gaussian NB and SGD for X_val.
- Evaluate the blended model's performance on the validation dataset (X_val, y_val)
- Perform hyperparameter tuning for the neural network and SGD (momentum, weight_decay, etc.).
- Train the blended model using both X_train and X_val, incorporating the optimal hyperparameters.
- Assess the generalization performance of the final model on the test dataset (X_test, y_test).

The algorithm outlines a comprehensive approach for blending Gaussian Naive Bayes with SGD while incorporating early stopping for machine learning tasks. Combining Stochastic Gradient Descent (SGD) with Gaussian Naive Bayes for large-scale linear models leverages both approaches' advantages. Although it performs well on normally distributed data, Gaussian Naive Bayes has trouble with complicated datasets. SGD effectively optimizes parameters, improving performance and accuracy. Gaussian Naive Bayes is the method used in this technique to estimate initial probability. The initial step in preparing the training, validation, and test datasets is data preparation, which involves feature encoding and scaling. GNB and SGD are combined through steps that include initializing the GNB model, producing probabilistic outputs, configuring the SGD classifier, integrating GNB probabilities with the original features, training the SGD classifier, and forecasting the last prediction stage. This guarantees correctness and precision in training and estimating class probabilities. The algorithm then moves forward with two crucial parts. First, it builds a Gaussian NB model on the training data and uses it to make probabilistic forecasts on the verification dataset. To carry out prompt halting and avoid overfitting, it then creates and trains a neural network using SGD while monitoring a validation metric [22].

After these training phases, the programme then integrates the forecasts from both Gaussian NB and SGD. It assesses the performance of the blended model on the validation data and enables hyperparameter adjustment, optimizing both the SGD and neural network design. A blended model that combines probabilistic predictions with iterative optimization skills is enhanced by combining Gaussian Naive Bayes and Stochastic Gradient Descent (SGD). This method improves generalization by utilizing the speed and robustness of Naive Bayes to decrease bias and alter decision boundaries more precisely. Based on input characteristics, the computer predicts malware and benign classes using GNB and stochastic gradient descent (SGD) models. The GNB model generates probabilistic predictions, while SGD combines raw characteristics and GNB probabilities to improve prediction accuracy. The approach ends by training the final blended model across the entire training dataset and evaluating its generalization performance across the test dataset. It uses a structured method to improve classification tasks by combining probabilistic and deep learning approaches, emphasising documentation, interpretation, and discussions about future research paths [23]. Ensemble Learning is a systematic technique to enhance classification jobs by fusing deep learning and probabilistic methods. Creating a robust predictive model entails integrating predictions from several models, such as deep neural networks and Gaussian Naive Bayes. This approach uses deep learning's capacity to recognize intricate patterns and probabilistic reasoning's advantages.

Gaussian NB, a probabilistic classifier suited for continuous data, can be a foundational component. SGD, an optimization algorithm, updates model parameters iteratively to minimize loss functions. Malware detection systems require constant upgrading and monitoring to stay updated with the newest threats. Real-time network traffic, system behaviour, and user activity analysis are all part of this process, including machine learning models, patches, and threat intelligence feeds. Weight and bias fixing methods can stabilize and optimize the training process, ensuring convergence without numerical issues. Implement feature scaling, regularise strategies like L2 regularisation, adjust the learning rate, pre-process input data, and utilize suitable initialization approaches for model parameters in SGD to guarantee the best possible performance for Gaussian NB and SGD optimization. These procedures ensure a solid foundation for excellent performance and avoid numerical problems. Early stopping, a regularization technique, monitors validation metrics during training and halts the process if performance plateaus or deteriorates, preventing overfitting. This combination offers a versatile model capable of handling mixed data types and efficiently optimizing training while safeguarding against overfitting [24].

4. RESULTS AND DISCUSSION

4.1 Performance analysis of the proposed system for the Malware dataset

This balanced dataset has 100,000 samples, split evenly between malware and benign classifications. It likely contains hash values, millisecond timestamps, process classifications, states, usage counters, priorities, and policy information, reflecting various aspects of system processes. The balanced distribution suggests an intentional effort to create a dataset for binary classification tasks, particularly in the context of malware detection [9, 25]. Reduced bias and improved learning are two ways a balanced dataset improves binary classification model performance. Resampling methods, feature engineering, stratified sampling, and synthetic data creation are used to attain this equilibrium. In addition to enhancing generalization and lowering the possibility of overfitting the dominant class, these approaches guarantee that all classes receive equal attention.

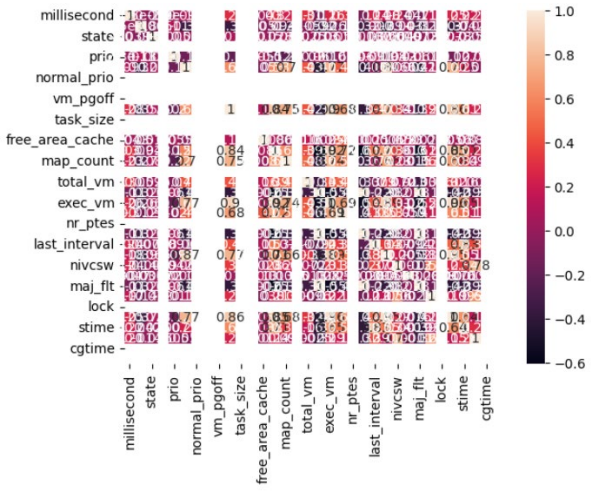

Figure 2. Correlation Matrix of malware dataset

The StndardScaler optimization procedure ensures that all features have a comparable scale, which is essential for the efficiency of many machine learning algorithms. StandardScaler improves model performance by maintaining associations between data points within each feature, especially for algorithms sensitive to feature scales [26]. In addition to preserving feature associations, optimizing model performance, lowering bias, and enhancing compatibility with techniques such as logistic regression, SVMs, and neural networks, Standard Scaler standardizes features by eliminating mean and scaling to unit variance. It increases model accuracy, lessens bias towards features with broader ranges, and maintains relative distances and relationships between data points. This processing phase is crucial for various machine learning applications to produce steady and reliable forecasts. Figure 2 contains the malware dataset's correlation matrix. According to Figure 2 [27], each matrix cell shows a correlation coefficient, which can be between -0.6 and 1.

Seven distinct layers comprise the model: a single input layer with 27 units, six hidden levels with fifty elements each, a single output level with two units, and a SoftMax activation layer for the classification task. Rectified Linear Unit (ReLU) is the activation function employed for the buried layers. Through several hidden layers, this design enables the model to understand complicated correlations in the data. To make decisions, each layer's neurone count is decreased, hidden layers' ReLU is used for effective gradient propagation, and a sigmoid function in the output layer represents probabilities. The model summary explains its architecture, including the number of trainable parameters in each layer, making evaluating and improving the model architecture easier while training [28]. Equation 5 is used to calculate the model's accuracy.

$$A = \frac{correctlypredicted}{Numberofcorrect \wedge incorerctsamples} \quad (5)$$

Table 2. Summary of the Dense Layer Architecture

Layer (type)	Output Shape	Parameter
dense	50	1400.0
dense_1	50	2550.0
dense_2	40	2550.0
dense_3	30	2550.0
dense_4	20	2550.0
dense_5	10	2550.0
dense_6	2	102.0

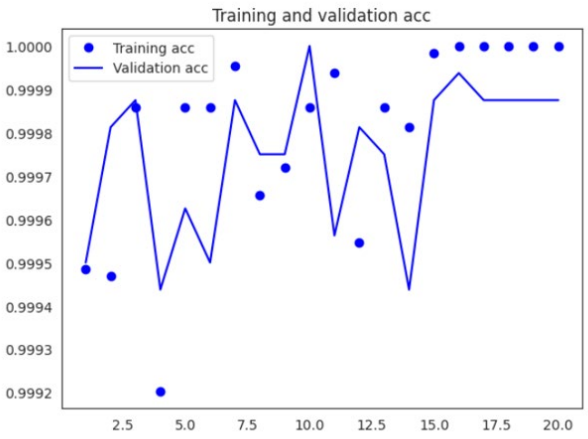

Figure 3. Accuracy of proposed work without early stopping

An overview of the neural network algorithm's dense layer design may be seen in Table 2. The type, input shape, and number of parameters that can be trained for each layer are

listed in the table. A total of seven layers make up the model: the input layer and six layers that are hidden. Each hidden layer has 50 units (neurons), and the final output layer consists of 2 units for classification purposes. The total number of parameters for each hidden layer is 2550, while the input layer has 1400 parameters. The output layer has 102 parameters. This architecture allows the model to capture complex patterns and relationships within the data, making it suitable for various machine-learning tasks [29].

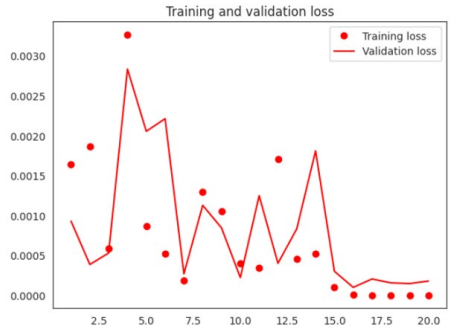

Figure 4. Loss of proposed work without early stopping

The provided output in Figures 3 and 4 reveals the evaluation results of a machine learning model, indicating exceptional performance. The training loss, an indicator of prediction accuracy on the training data, is impressively low at 7.6280e-04, accompanied by a training accuracy of 99.99%, signifying accurate predictions on the training dataset. The model's ability to generalize to previously unknown data is impressive, as shown by the test's low-test loss of 0.000763 and its high-test accuracy of 99.989998%. Although the results are promising, overfitting is an issue. There are more local minima than elsewhere. While the findings show excellent accuracy and loss measures, it's important to note that overfitting may be a risk, mainly if the model performs considerably better on the training dataset than the test dataset. Overfitting happens when a model learns to fit the training data too closely. This captures noise and particular patterns that might not generalize effectively to fresh data [30].

Table 3 presents the performance results of a neural network model under various hyperparameter configurations. The hyperparameter tuning process includes SGD hyperparameters like batch size, regularisation term, and learning rate and Gaussian NB hyperparameters like the smoothing parameter, which regulates the degree of the prior probability adjustment. Cross-validation adjusts these parameters to balance variance and bias, guaranteeing model correctness and training efficiency. Cross-validation in conjunction with grid search or random search is used to meet the selection criterion. Each row corresponds to a specific setup, featuring different learning rates and optimizers. For instance, in Row 1, with a learning rate of 0.01 and the SGD optimizer, the model achieved a training loss of 0.0023, indicating a solid fit for the training data with a training accuracy of 99.2%. Similarly, the test loss (0.0031) and test accuracy (98.7%) suggest good generalization to unseen data. The table showcases how these hyperparameters influence the model's ability to learn and generalize, aiding in selecting the most effective configuration for specific machine-learning tasks [31].

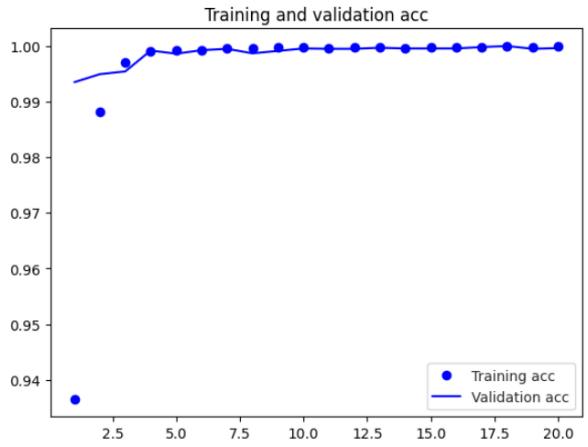

Figure 5. Training and validation accuracy of proposed PDE with early stopping

Table 3. The performance analysis of PDE for different optimizers and learning rate

Learning Rate	Optimizer	Training Loss	Training Accuracy	Test Loss	Test Accuracy
0.01	SGD	0.0023	99.2%	0.0031	98.7%
0.001	Adam	0.0012	99.5%	0.0015	99.0%
0.1	RMSprop	0.0041	98.8%	0.0053	98.3%
0.005	Adam	0.0019	99.3%	0.0022	99.1%
0.03	SGD	0.0036	98.9%	0.0042	98.6%

Figure 6. Training and validation loss of proposed PDE with early stopping

The output in Figures 5 and 6 represents the evaluation results of a machine learning model, showcasing its exceptional performance. The training loss, a measure of

prediction accuracy on the training data, is shallow at 6.6912e-04, with a training accuracy of 99.98%, indicating that the model accurately predicts the training dataset. The model's generalization to new, unseen data is equally impressive, as reflected in the low-test loss of 0.000669 and a test accuracy of 99.98%. These results demonstrate the model's robust training and ability to make accurate predictions without overfitting and local minima [32]. An optimization of a model that has several local minima may result in poor generalization to unknown data, overfitting, and decreased performance. Stochastic Gradient Descent (SGD), regularisation methods, early halting, adaptive learning rates, and merging several models via bagging or boosting are some strategies to deal with the issue. By using these strategies, bad local minima may be avoided, and the test data performance of the model will be enhanced.

4.2 Performance analysis of the proposed system for the UCI Malware dataset

The dataset comprises 373 samples from Windows executables, encompassing 301 malicious files and 72 non-malicious files. Each sample is characterized by 531 features, which include a combination of binary hexadecimal representations and Dynamic Link Library function calls. The label column indicates whether a file is categorized as malicious or non-malicious, as shown in Figure 7. Behavioural features like system calls, memory usage, and network activity are combined with metadata like file origin, timestamp, and digital signatures, and static features like file properties like size, type, entropy, and hash values to create the UCI Malware dataset, a comprehensive tool for malware detection. These characteristics aid in detecting complex malware, assist in identifying well-known malware patterns, and serve as a foundation for machine learning models trained to discriminate between benign and dangerous software accurately.

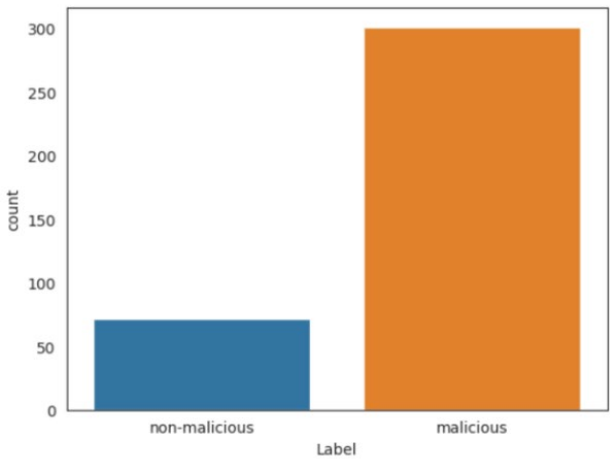

Figure 7. Malicious and non-malicious data

Table 4: Performance of machine learning model for uci-malware dataset

Model	Accuracy (%)
RandomForestClassifier	0.893
DecisionTreeClassifier	0.907
KNeighborsClassifier	0.872
AdaBoostClassifier	0.915
SGDClassifier	0.884
ExtraTreesClassifier	0.901
GaussianNB	0.835
Probabilistic descent ensemble	0.99

The classifiers mentioned above encompass a diverse set of machine-learning algorithms. RandomForestClassifier utilizes ensemble learning and multiple decision trees to enhance predictive accuracy. DecisionTreeClassifier constructs a tree-like structure to make decisions based on input features. KNeighborsClassifier relies on the majority class of nearby data points in feature space for classification. AdaBoostClassifier combines weak learners to form strong ones, focusing on previously misclassified samples. In situations involving complicated decision-making, AdaBoostClassifier performs better than KNeighborsClassifier. By concentrating on difficult instances, iteratively modifying misclassified samples enhances model performance and builds a robust ensemble model that performs better with various intricate datasets. AdaBoost is beneficial when model interpretability is less critical, including outliers or overlapping classes. SGDClassifier employs Stochastic Gradient Descent for optimization, commonly used for linear classifiers. ExtraTreesClassifier is akin to RandomForest but introduces feature selection randomness. Lastly, GaussianNB, a type of Naive Bayes classifier, assumes a normal distribution of features and is often employed in classification, particularly in text categorization. These classifiers cater to various machine learning tasks, offering versatility in handling different datasets and challenges [33].

The accuracy-based performance of different machine learning models on the UCI-Malware dataset is shown in Table 4. A variety of classifiers, including RandomForest, DecisionTree, KNeighbors, AdaBoost, SGD, ExtraTrees, and GaussianNB, are included in the table. These models' accuracy scores result from training and testing them on the dataset. The data in the table show how each model fared on the challenge, while accuracy is a statistic that shows the percentage of correctly categorized examples. The AdaBoostClassifier, which had an accuracy of 0.915, came in first in the results, closely followed by the DecisionTreeClassifier and ExtraTreesClassifier, which had accuracy ratings of 0.907 and 0.901, respectively. The Probabilistic Descent Ensemble surpasses or is roughly on par with other sophisticated algorithms in a particular machine training task, as seen in Figure 8, where it obtains a precision of 0.99% compared to other cutting-edge models [33, 34].

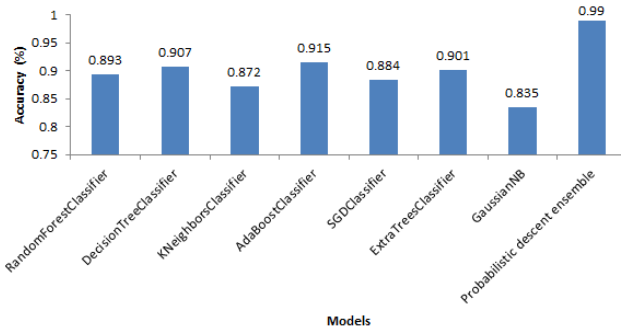

Figure 8. Performance analysis of proposed work with other models

5. CONCLUSION

Deep learning's Probabilistic Descent Ensemble (PDE) technique offers a viable way to enhance virus prediction. This innovative method harnesses the collective strength of multiple neural network models with diverse architectures and training strategies, yielding superior accuracy while mitigating the risk of false positives. Combining Stochastic Gradient Descent (SGD) with early stopping is a robust optimization approach, effectively preventing overfitting by monitoring validation metrics during training. This synergy between SGD and early stopping enhances model performance. It makes it adaptable to various tasks and datasets, potentially reducing training time and improving generalization capabilities.

Moreover, incorporating a Gaussian Naive Bayes (NB) classifier with SGD optimization enriches the model's probabilistic framework, catering to data distributions that follow a Gaussian pattern. The achieved 99% accuracy in malware prediction underscores the effectiveness of this approach, demonstrating its resilience against overfitting and local minima. In summary, this research opens the door for more robust and effective malware detection systems, potentially advancing cybersecurity significantly.

Declarations

Funding: None of the authors received funds or grants.

Conflict of interest: There is no conflict of interest among the authors.

Data Availability: All data generated or analyzed during this study are included in the manuscript.

Code Availability: Not applicable.

Author's contributions: R. Vinoth Kumar and R. Suguna is contributed to the design and methodology of this study, the assessment of the outcomes and the writing of the manuscript.

References

[1] Gavriluţ D, Cimpoeşu M, Anton D, Ciortuz L. Malware detection using machine learning. 2009 International multiconference on computer science and information technology. 2009;735-741.

[2] Xu Z, Ray S, Subramanyan P, Malik S. Malware detection using machine learning based analysis of virtual memory access patterns. Des Autom Test Eur Conf Exhib. 2017;169-174.

[3] Naway A, Li Y. A review on the use of deep learning in android malware detection. arXiv preprint arXiv:1812.10360. 2018.

[4] Gorment NZ, Selamat A, Cheng LK, Krejcar O. Machine learning algorithm for malware detection: Taxonomy, current challenges and future directions. IEEE Access. 2023.

[5] Rathore H, Agarwal S, Sahay SK, Sewak M. Malware detection using machine learning and deep learning. In: Big Data Analytics: 6th International Conference, BDA 2018, Warangal, India, December 18–21, 2018, Proceedings 6. Springer Int Publ. 2018;402-411.

[6] Gavriluţ D, Cimpoeşu M, Anton D, Ciortuz L. Malware detection using machine learning. 2009 International multiconference on computer science and information technology. 2009;735-741.

[7] Liu L, Wang BS, Yu B, Zhong QX. Automatic malware classification and new malware detection using machine learning. Front Inf Technol Electron Eng. 2017;18(9):1336-1347.

[8] Mahindru A, Sangal AL. MLDroid—a framework for Android malware detection using machine learning techniques. Neural Comput Appl. 2021;33(10):5183-5240.

[9] Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Venkatraman S. Robust intelligent malware detection using deep learning. IEEE Access. 2019;7:46717-46738.

[10] Karbab EB, Debbabi M, Derhab A, Mouheb D. MalDozer: Automatic framework for android malware detection using deep learning. Digit Investig. 2018;24

[11] Urooj U, Al-rimy BAS, Zainal A, Ghaleb FA, Rassam MA. Ransomware detection using the dynamic analysis and machine learning: A survey and research directions. Appl Sci. 2021;12(1):172.

[12] Firdausi I, Erwin A, Nugroho AS. Analysis of machine learning techniques used in behavior-based malware detection. 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies. 2010;201-203.

[13] Kumar MS, Purusothaman T, Kumar RL. Secure and reliable intrusion detection scheme for software-defined networking using LFTS-Rnn and PC-JTFOA. IETE J Res. 2024;1-16.

[14] Punitha P, Kumar L, Revathi S, Premalatha R, Aiswarya RS. Secured framework with a hash function-enabled keyword search in cloud storage services. Int J Coop Inf Syst. 2024;2450001.

[15] Lakshmana Kumar R, Jayanthi S, Muthu B, Sivaparthipan CB. An automatic anomaly application detection system in mobile devices using FL-HTR-DBN and SKLD-SED K means algorithms. J Intell Fuzzy Syst. 2023;(Preprint):1-14.

[16] Muthu B, Cb S, Kumar PM, Kadry SN, Hsu CH, Sanjuan O, Crespo RG. A framework for extractive text summarization based on deep learning modified neural network classifier. ACM Trans Asian Low-Resour Lang Inf Process. 2021;20(3):1-20.

[17] Dataset Collection: https://www.kaggle.com/code/vinesmsuic/malware-detection-using-deeplearning/input.

[18] Dataset Collection: https://www.kaggle.com/code/maidaly/malware-detection-with-machine-learning/input.

[19] Mahindru A, Sangal AL. SOMDROID: Android malware detection by artificial neural network trained using unsupervised learning. Evol Intell. 2022;15(1):407-437.
[20] Shaukat K, Luo S, Varadharajan V. A novel deep learning-based approach for malware detection. Eng Appl Artif Intell. 2023; 122:106030.
[21] Venkatraman S, Alazab M, Vinayakumar R. A hybrid deep learning image-based analysis for effective malware detection. J Inf Secur Appl. 2019; 47:377-389.
[22] Raymond VJ, Raj RJR, Retna J. Investigation of android malware with machine learning classifiers using enhanced PCA algorithm. Comput Syst Sci Eng. 2023;44(3):2147-2163.
[23] Udayakumar N, Saglani VJ, Cupta AV, Subbulakshmi T. Malware classification using machine learning algorithms. 2018 2nd International Conference on Trends in Electronics and Informatics. 2018;1-9.
[24] D'Angelo G, Ficco M, Palmieri F. Malware detection in mobile environments based on autoencoders and API-images. J Parallel Distrib Comput. 2020; 137:26-33.
[25] Shhadat I, Hayajneh A, Al-Sharif ZA. The use of machine learning techniques to advance the detection and classification of unknown malware. Procedia Comput Sci. 2020; 170:917-922.
[26] Gupta SK, Pattnaik B, Agrawal V, Boddu RSK, Srivastava A, Hazela B. Malware detection using genetic cascaded support vector machine classifier in internet of things. 2022 Second International Conference on Computer Science, Engineering and Applications. 2022;1-6.
[27] Shaukat K, Luo S, Chen S, Liu D. Cyber threat detection using machine learning techniques: A performance evaluation perspective. 2020 International Conference on Cyber Warfare and Security. 2020;1-6.
[28] Aljabri M, Mirza S. Phishing attacks detection using machine learning and deep learning models. 2022 7th International Conference on Data Science and Machine Learning Applications. 2022;175-180.
[29] Selvaganapathy S, Nivaashini M, Natarajan H. Deep belief network-based detection and categorization of malicious URLs. Inf Secur J Glob Perspect. 2018;27(3):145-161.
[30] Alwaghid AF, Sarkar NI. Exploring malware behavior of webpages using machine learning technique: An empirical study. Electronics. 2020;9(6):1033.
[31] Masum M, Nur I, Faruk MH, Adnan M, Shahriar H. A comparative study of machine learning-based autism spectrum disorder detection with feature importance analysis. In: COMPSAC 2022: Computer Software and Applications Conference. 2022;3.
[32] Syafaâ L, Zulfatman Z, Pakaya I, Lestandy M. Comparison of machine learning classification methods in hepatitis C virus. Jurnal Online Inform. 2021;6(1):73-78.
[33] Htwe CS, Thant YM, Thwin MMS. Botnets attack detection using machine learning approach for IoT environment. J Phys Conf Ser. 2020;1646(1):012101.
[34] Rbah Y, Mahfoudi M, Balboul Y, Fattah M, Mazer S, Elbekkali M, Bernoussi B. Machine learning and deep learning methods for intrusion detection systems in IoMT: A survey. 2022 2nd International Conference on Innovative Research in Applied Science, Engineering and Technology. 2022;1-9.