WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

AUTOMATED CONFIGURATION OF DETECTION MODEL FOR REAL TIME MALWARE DEFENCE

Shakira Banu L[1], Suganraj P[2], Sundarapandiyan T[3], Surya Kumar R[4],Santhosh R[5]

Assistant Professor[1] , UG Student[2,3,4,5]

Department of Artificial Intelligence And Data Science, Dhanalakshmi Srinivasan Engineering College Perambalur, Tamil Nadu, India

saravanan.sharoz1772016@gmail.com[1], suganraj1828@gmail.com[2],sundardeva582003@gmail.com[4], rsksurya2003@gmail.com[3], santhoshrathinavel2001@gmail.com[5]

ABSTRACT:

In today's digital age, organizations themselves try to acquire user's data to make well-informed decisions at the edge. This increases the need for a solution that not only focuses on user privacy but also leverages the user's data. Federated Learning (FL) proves to be an effective solution for this purpose, but such techniques are dominated by sophisticated, resource-hungry Neural Network (NN) models. This makes it essential to create a FL technique that preserves user privacy while using light weight models. In the work that is proposed, we have established a new technique that combines FL with Support Vector Machines (SVM), which is a light weight traditional machine learning method. The initial SVM formulation minimizes the classification problem to a quadratic programming problem. This problem is then solved through techniques like Sequential Minimal Optimization (SMO), which forms the training process. Nevertheless, this training method for SVM is not natively supported by Federated Learning. To mitigate this, we reduced the hinge loss of SVM through Stochastic Gradient Descent (SGD) that yields the optimal SVM model supported by FL. The approach utilizes a meta-trained controller that dynamically determines the best learning rate for each epoch, unlike the fixed learning rate of traditional SGD. Empirical results show that the traditional SVM model incorporated with FL yields efficient outcomes similar to those of NN-based FL methods.

INTRODUCTION:

Just as crude oil came to be known as "black gold" during the Industrial Revolution, today's data is regarded as "digital gold." Businesses attempt to capture user data, which powers machine learning (ML), a mainstay of contemporary technology. Conventional ML is based on centralized data, and with this comes privacy concerns. Federated Learning (FL) rectifies this by training models on numerous devices without the sharing of user data—model updates are shared only. Whereas FL usually employs neural networks, their complexity is resource-demanding. Smaller and less complex Support Vector Machines (SVMs) are a possible alternative. Popular SVM implementations such as sklearn.svm.SVC, however, do not scale well ($O(N^2)$) and thus are inappropriate for big datasets. SGDClassifier was faster than SVC both in accuracy and scalability in experiments. Additionally, the learning rate in Stochastic Gradient Descent (SGD) has a significant impact on performance. Our results show that dynamically adjusting the learning rate improves model accuracy. With Bayesian optimization and SMAC3, we have experimented with static and dynamic learning rate tactics and concluded that dynamic methods are better. Our contributions are training an SVM through FL with adaptive control of learning rate, testing its meta-learning capacity in centralized and federated environments, and providing system architecture and implementation for this method.

EXISTING SYSTEM:

The current federated learning systems heavily rely on deep learning models like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for distributed training. Though effective, they are computationally intensive and not applicable to edge devices with restricted processing power and memory. Classical federated learning frameworks like FedAvg work with constant hyperparameters like learning rate, which typically results in sub-optimal convergence, particularly in non-IID data settings. These models demand regular interaction between clients and a master server, which adds latency and computational expense. Although Support Vector Machines (SVMs) are simple and efficient, few SVMs are incorporated into FL owing to their quadratic complexity ($O(N^2)$) and limited support for distributed systems. Moreover, few current systems do not leverage adaptive learning methods that dynamically tune hyperparameters according to the training scenario. This leads to increased training time, unstable convergence, and decreased performance in heterogeneous data environments. While some studies have attempted adaptive hyperparameter tuning with approaches such as SMAC3 and reinforcement learning, they are usually used in single-standalone environments and not effectively integrated into federated learning environments. As a result, the existing systems are not optimized to their full potential for privacy-preserving, scalable, and resource-constrained FL deployments.

DRAWBACKS:

1. Fixed Learning Rates: Fixed hyperparameters such as learning rate cause slow or unstable convergence.
2. Heavy Computational Load: Deep learning models demand high processing capacity, restricting usage on edge devices.
3. SVM Scalability Issues: Classical SVMs are not efficient for big data because of quadratic time complexity.
4. FL Incompatibility: Regular SVMs are not compatible with federated environments, restricting secure distributed usage.
5. Non-IID Data Challenge: Most FL systems suffer when client data is non-identically distributed.
6. High Communication Overhead: Frequent client-server interactions lead to increased latency and resource consumption.
7. No Adaptive Tuning: Dynamic hyperparameter tuning during training is not supported in most frameworks.
8. Steep Learning Curve: Complicated setup and tuning make systems challenging for novice developers.
9. Limited Client Resources: Current models disregard the low computing capabilities of edge devices.
10. Rigid Model Design: Present systems are frequently bound to certain models, decreasing flexibility.

PROPOSED SYSTEM :

The suggested system presents a compact, light-weight, and privacy-conscious federated learning model for malware detection via dynamic analysis of system-level activity. In contrast to conventional FL models based on deep neural networks, this scheme takes advantage of the low-resource and simplicity requirements of Support Vector Machines (SVMs) and is, therefore, particularly well-suited for edge devices. One of the major innovations of this system is the inclusion of a dynamic learning rate controller at the client level. Rather than employing a static learning rate, every client dynamically adjusts its own learning rate during local training based on gradient-based metadata. This controller is trained with SMAC3 (Sequential Model-based Algorithm Configuration), a fast hyperparameter optimization method, enabling the model to learn in real-time to changing data and training environments. Each client trains its local model with Stochastic Gradient Descent (SGD) and sends updates to a central server, which combines them to construct a global model through weighted averaging.

This architecture overcomes some of the current limitations in federated learning, such as convergence instability, fixed hyperparameters, and high resource consumption. The dynamic tuning approach enhances model accuracy and training efficiency even in non-IID settings, which are typical in real-world distributed systems. By employing SVMs rather than deep models, the system greatly decreases computation and memory usage, and deployment becomes possible on low-power devices. The system also comes with a Flask-based web interface through which users can securely login, input behavioral data, and get real-time malware predictions. All inputs and predictions are traced to both a database and a CSV file for traceability. In summary, the system presented here shows a realistic, scalable, and smart solution to federated malware detection, striking a balance between accuracy, privacy, and computational expense—providing a feasible solution for protecting distributed environments from evolving cyber threats.

ADVANTAGES :

1. Privacy-Preserving Learning: Sensitive information is kept on client devices, promoting user privacy via federated learning.
2. Lightweight Model Architecture: Leverages SVMs in place of deep neural networks, minimizing computational and memory needs.
3. Efficient on Edge Devices: Optimized for low-resource environments, making it perfectly suited for real-world edge deployment.
4. Handles Non-IID Data: Works optimally even if client data distributions vary, which is a frequent problem in federated environments.
5. Scalable Framework: Allows for easy addition of more clients or enlarging datasets without rebuilding the architecture.
6. Real-Time Prediction Interface: The built-in Flask web application enables users to enter data and receive real-time malware predictions.
7. Dual Logging System: Stores predictions in both a database and CSV file, making it traceable and auditable.
8jg. Flexible Integration: Extendible with other models, new features, or real-time data collection tools in future releases..

SYSTEM ARCHITECTURE:

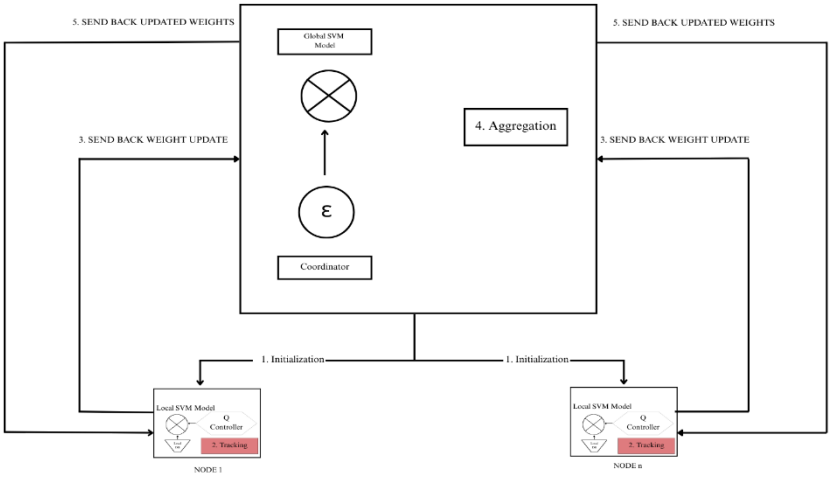

Fig 1. System Architecture

LIST OF MODULES :

1. Federated Learning Client
2. Step-Size Controller
3. Stochastic Gradient Descent (SGD)
4. Federated Learning Server
5. Weight Aggregator

MODULE DESCRIPTION :

1. Federated Learning Client

The Federated Learning Client is responsible for training local SVM models directly on user devices, maintaining data privacy by not sharing raw data. Each client processes their data independently, contributing model updates to a global model. This decentralized training approach ensures privacy and security by keeping data local to the device. The model updates are aggregated periodically by the server. The local training results in models tailored to specific device data, improving performance. This method enables collaborative learning while safeguarding sensitive information.

2. Step-Size Controller

The Step Size Controller adjusts the learning rate for every client adaptively by using a meta-learning solution, enhancing convergence rate. By learning the best learning rate for each client's improvement, it raises the stability of training. The adjustment averts abrupt large updates that could destabilize the model. The controller makes the system adjustable to individual client requirements, optimizing training efficiency. It optimizes learning dynamics, improving the overall training process . This module achieves quicker and more stable convergence of models across clients.

3. Stochastic Gradient Descent (SGD)

The Stochastic Gradient Disorder module introduces controlled variation in gradient updates using Stochastic Gradient Descent (SGD). This randomness in the updates helps improve the robustness of the model, allowing it to generalize better to new data. By preventing overfitting, it enhances the model's adaptability to variations in the data. The randomness helps the model escape local minima, improving the overall optimization process. This method also ensures the model is not overly fitted to the training data. Consequently, it supports better performance when the model encounters unseen data.

4. Federated Learning Server

The Weight Aggregator combines client contributions through weighted averaging, ensuring fair representation of each client's update. The aggregation process factors in the performance or data size of each client to determine how much weight their contribution carries. By using weighted averaging, it ensures that the global model reflects a balanced mix of all client inputs. This prevents the model from being biased toward any single client’s data. The Weight Aggregator helps create a robust global model, improving overall model performance. It ensures that the global model benefits from the diversity of client data.

5. Weight Aggregator

The Weight Aggregator aggregates client contributions using weighted averaging, which guarantees proportionate representation of any client's update. The process of aggregation takes into consideration the performance or the data size of each client and how much weight their contribution will be assigned. Using weighted averaging, the global model captures a balanced representation of all the client inputs. This ensures that the model cannot be biased toward one client's data in particular. Weight Aggregator aids in the development of the reliable global model with enhanced overall model performance. The global model is guaranteed to benefit from client data diversity.

RESULT:

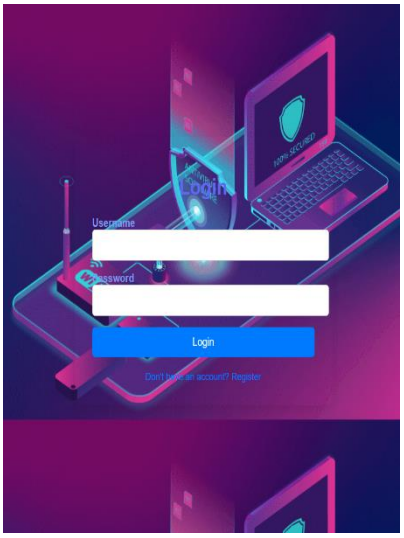

Fig.2

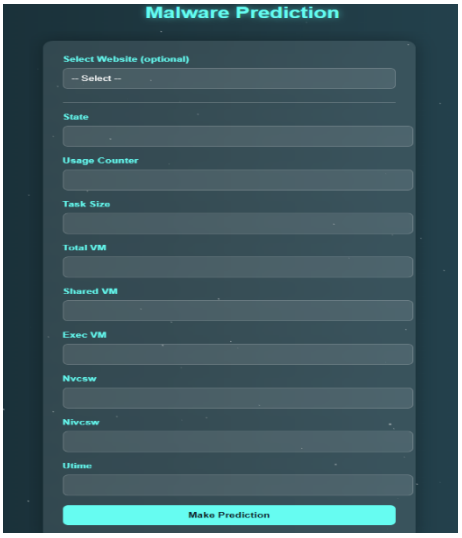

Fig.3

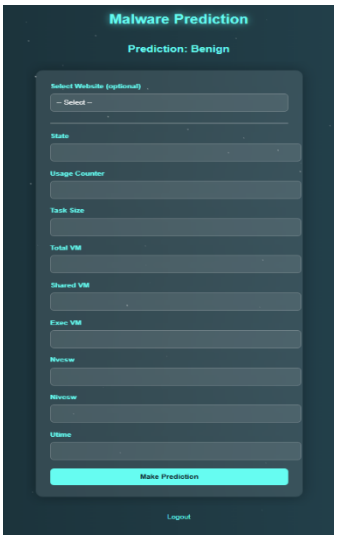

Fig.4

CONCLUSION:

The project effectively deploys an automated malware detection system through a dynamically configurable machine learning model in a real-time analysis environment. Through the combination of a Flask-based web application, a trained global model, and a user-friendly interface supported by a SQLite database, the system provides efficient, scalable, and accessible malware protection. The application of dynamic analysis improves detection precision by monitoring runtime behavior, while the automated configuration process enables the model to learn to adapt to various input patterns with minimal human intervention. This method provides timely and precise identification of malicious activity, which helps to improve cybersecurity in real-world settings.

FUTURE ENHANCEMENT:

To further tighten the system, future research can aim to include more sophisticated ensemble learning methods or deep learning frameworks for better detection accuracy and robustness. Real-time threat feeds can be used to refresh the detection model with the most current malware signatures and behavior patterns. Adding sandboxing or virtualization support to the dynamic analysis environment will enable safer and more granular behavioral analysis. Multi-platform malware detection support, such as Windows, Linux, and mobile platforms, can enhance the coverage of the system. Lastly, integration of a centralized dashboard with analytics, alert management, and automated response capabilities would enhance usability and efficiency in enterprise settings.

REFERENCES:

1. Abadi, M., et al., TensorFlow: Large-scale machine learning on heterogeneous systems, TensorFlow.org, Version 2.x, 2024.
2. TFF Authors, TensorFlow Federated: Machine Learning on Decentralized Data, TensorFlow.org, 2024.
3. Daniel, C., et al., Learning step size controllers for robust neural network training, AAAI Conference on Artificial Intelligence, 2023.
4. Biedenkapp, A., et al., Dynamic Algorithm Configuration: Foundation and Applications, NeurIPS, 2022.
5. Caldas, S., et al., LEAF: A Benchmark for Federated Settings, arXiv preprint arXiv:1812.01097, 2022.
6. Chang, C.-C., & Lin, C.-J., LIBSVM: A Library for Support Vector Machines, ACM Transactions on Intelligent Systems and Technology, Vol. 2, No. 3, 2019.
7. Cohen, G., et al., EMNIST: Extending MNIST to handwritten letters, Proceedings of IJCNN, 2019.
8. Deng, L., The MNIST Database of Handwritten Digit Images, IEEE Signal Processing Magazine, 2018.
9. Eimer, T., et al., DACBench: A Benchmark Suite for Dynamic Algorithm Configuration, NeurIPS, 2016.
10. Fan, R.-E., et al., LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research, Vol. 9, 2015Viola, P., & Jones, M. (2001).