Federated Hyperdimensional Ensembles for Mobile Malware Detection

Andrea Augello[1,*], Alessandra De Paola[1], Giuseppe Lo Re[1] and Gioacchino Zangara[1]

[1] *Department of Engineering, University of Palermo, Italy*

Abstract

Mobile devices face an ever-increasing threat from cyber attacks, with malware constantly evolving to circumvent traditional detection systems. Adaptive techniques attempt to address this problem by retraining models through updated data, but this approach requires sharing sensitive user information, raising significant privacy concerns. This work introduces a novel federated learning architecture for mobile malware detection to address these challenges. The proposed approach integrates the efficiency of Hyperdimensional Computing (HDC) to create an ensemble of lightweight models. The federated architecture enables collaborative model training across multiple devices without requiring users to share their private data, thus preserving privacy. The architecture supports heterogeneous clients, allowing devices with different computational capabilities to participate effectively in the training process. Experimental evaluations demonstrate the system's effectiveness in maintaining high performance in a privacy-preserving manner, even in dynamic, resource-constrained contexts.

Keywords

Federated learning, Hyperdimensional Computing, Mobile Malware Detection

1. Introduction

Mobile devices are increasingly targeted by cyber-attacks due to their expanding attack surface. The rapid evolution of malware, driven by the emergence of new vulnerabilities, results in constantly changing attack patterns. [1]. Traditional signature-based methods [2], while lightweight, are easily bypassed by code modifications, polymorphic techniques, or obfuscation through compression. To counter these techniques, research has focused on machine learning for automated malware detection [3].

However, the evolution of malware makes traditional malware detection systems ineffective over time [4]. This phenomenon, known as concept drift [5], refers to the change in statistical properties of data between the training phase and online operation. This problem is critical in many cybersecurity applications [6, 7, 8, 9], and there is actually no universal solution to address it. A widely adopted approach is to periodically retrain models to adapt to the latest data distributions and maintain their performance [4]. Such an approach is challenging in the mobile context, where threats evolve rapidly and devices often lack the computational resources to support frequent updates [10]. A straightforward solution to maintain an updated model would involve periodically collecting the applications installed by each user in a central server, where a model can be using the aggregated up-to-date data. Collecting this data often requires users to share personal information, raising significant privacy and security concerns and requiring a high level of trust in the server, and thus it is not a viable approach.

Federated Learning (FL) has emerged as a potential solution to preserve user privacy, allowing collaborative model training among multiple users without sharing raw data [11, 12]. Instead of sharing training data, clients transmit only model weights, preserving user privacy while still allowing a global model to adapt effectively. FL can build a global model capable of integrating information from all clients and continuously updating itself, mitigating performance degradation caused by concept drift [13].

Such an approach thus results particularly suitable for cyber security scenarios [14] where it is necessary to mitigate performance degradation due to evolving data without compromising user privacy.

Ital-IA 2025: 5th National Conference on Artificial Intelligence, organized by CINI, June 23-24, 2025, Trieste, Italy

[*] Corresponding author.

andrea.augello01@unipa.it (A. Augello); alessandra.depaola@unipa (A. De Paola); giuseppe.lore@unipa.it (G. Lo Re); gioacchino.zangara@unipa.it (G. Zangara)

0000-0001-9085-4218 (A. Augello); 0000-0002-7340-1847 (A. De Paola); 0000-0002-8217-2230 (G. Lo Re)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2025-12-07

However, FL presents limitations, particularly concerning the complexity of applicable models. Participating mobile devices may lack the computational resources required to train large and complex models. This limitation can hinder the effectiveness of the approach in situations where more sophisticated models are needed to achieve optimal performance; it also prevents resource-constrained users from participating in the training process and benefiting from the trained model.

This work proposes a distributed architecture capable of performing malware detection, while preserving privacy and limiting the computational load on users' devices. The solution is based on a federated approach where learning occurs locally on clients and models are aggregated centrally. The architecture uses an ensemble of models based on the Hyperdimensional Computing (HDC) paradigm, chosen for its efficiency and lightness. Furthermore, the architecture enables clients with heterogeneous computational capabilities to contribute according to their resources without compromising the overall quality of the model.

The remainder of this work is organized as follows. Section 2 discusses preliminary concepts. Section 3 details the proposed architecture. Section 4 presents the experiments. Section 5 concludes the paper.

2. Background on HDC

Hyperdimensional Computing (HDC) is a computational paradigm that uses high-dimensional vectors, called hypervectors (HVs), in order to represent information. Randomly chosen hypervectors tend to be mostly orthogonal to each other due to the high dimensionality of the space in which they are represented. This property is fundamental for the HDC paradigm, since it allows to combine different information in a way that minimizes interference between them. HDC is particularly well-suited for resource-constrained devices such as mobile devices due to its computational efficiency and low memory requirements since it relies on simple operations [15].

HDC encodes samples into high-dimensional HVs using an embedding matrix Φ. In this work, the rows in Φ are randomly and uniformly sampled from the surface of a unit sphere of dimension D, ensuring that the similarity relations in the input space are preserved in hyperspace [16]. Samples $\mathbf{X}$ are encoded into HVs $\mathbf{H}$ through a simple matrix multiplication: $\mathbf{H} = \mathbf{X}\mathbf{\Phi}^T$.

A key operation supported by the HDC paradigm is the element-wise addition (bundling) of multiple hypervectors. This operation allows the information contained in two hypervectors to be merged into a new vector, which preserves the contribution of both original vectors: the resulting hypervector is similar to the two operands but dissimilar from other hypervectors. This property can be leveraged to train classification models.

The HDC computational paradigm is based on the generation of class hypervectors C_l. Each hypervector represents a prototype for one of the l classes to be predicted. Class HVs are often computed as the sum of the HVs of the samples belonging to that class. Classification is performed by assessing the similarity $\delta(\cdot, \cdot)$, of a query sample to the class HVs. The class whose HV is most similar to the sample is assigned as a prediction. Usually, δ is the cosine similarity

$$\delta(H, C_l) = \frac{H \cdot C_l}{\|H\|_2 \times \|C_l\|_2}, \tag{1}$$

where $H \cdot C_l$ is the scalar product between the query hypervector H and the class hypervector C_l.

The predicted label for a sample i is thus $y_i' = \arg\max_j \delta(H_i, C_j)$.

The class HV are often further refined by a retraining step in which misclassified samples are used to further update the class HVs [17]. Instead of relying on a static encoding matrix during the training phase, where the initially generated dimensions remain unchanged throughout the process, some approaches implement dynamic encoding, in which less relevant features are periodically restored during the retraining phase to improve the effectiveness of the model [17].

HDC's computational efficiency makes it ideal for FL, enabling local training and global model updates through client-server exchanges [18]. Ensembles of HDC models have shown improved performance over single models [19]. However, ensembles with HDC in federated contexts is underexplored.

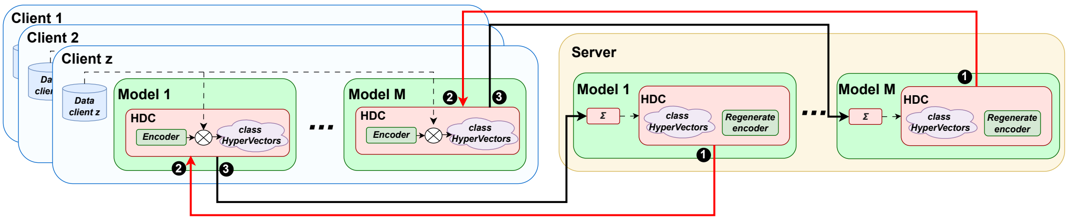


Figure 1: Structure of the system, with a focus on the client-server exchange involving the ensemble models. 1) clients receive an initial, possibly outdated, model from the server to be trained locally; 2) after training locally, each client sends the updated model weights to the server; 3) the server aggregates all the contributions and returns the updated model and possibly, a new encoding matrix.

3. System architecture

The architecture presented in this work employs an ensemble of HDC models trained by multiple clients within a FL framework, coordinated by a central server. Fig. 1 provides a high-level illustration of the system structure. Multiple clients, each with a private dataset, receive an initial ensemble of models from the server, each consisting of a set of class hypervectors $\mathbf{C}$ and an encoding matrix $\mathbf{\Phi}$. Since the models are independent, each client can train all the models in its ensemble independently in parallel, depending on the available computational power. In accordance with the HDC paradigm, over a few iterations, each client updates the class HVs according to the misclassified samples as follows:

$$C_l \mathrel{+}= \sum H_i[y_i = l, y_i' \neq l] - \sum H_j[y_j \neq l, y_j' = l]. \tag{2}$$

Each class HV is updated by bundling it with the encoded samples of the class l that were misclassified, increasing their similarity to C_l. Conversely the weighted HVs of the samples belonging to other classes that were misclassified as l are subtracted, making C_l less similar to them. After each update, the class HVs are sent to the server, which averages them across all clients to form the global class HVs.

To address concept drift, the models in the ensemble employ a periodic dimension regeneration mechanism (illustrated in Fig. 2). This involves identifying and replacing the least significant dimensions in the HDC encoding matrix $\mathbf{\Phi}$, enabling the model to adapt to new data patterns and maintain performance over time: some learned patterns may become irrelevant due to the evolution of the data distribution, and thus some of the model's representation capabilities can be better exploited by regenerating the dimensions involved.. The selection of the k dimensions to form the set $\mathcal{D}'$ for regeneration is done by identifying those components for which the distance to the incorrectly predicted class is small, while the distance to the correct class is high:

$$\operatorname*{arg\,min}_{|\mathcal{D}'|=k} \sum_{j\in\mathcal{D}'} \left(\sum_i \left(\|H_i - C^{y_i'}\|_1 - \|H_i - C^{y_i}\|_1\right)\right)_j , \tag{3}$$

where $\|H_i - C^{y_i}\|_1$ and $\|H_i - C^{y_i'}\|_1$ is the dimension-wise distance between the hypervector of the sample with the ground truth class HV and the prediction class HV respectively. Thus, dimensions that contributed to the incorrect prediction are identified. These dimensions are then set to zero in the class HVs and resampled from a normal distribution in the encoding matrix, so that further training can encode new information in the regenerated dimensions. When updating the HVs, the ensemble weigths are also updated. The entire training process is executed in a federated manner to ensure user data remains localized.

Clients interact with the server over multiple rounds in parallel for all the models in the ensemble:

1. For each model m, the server provides an initial encoding matrix $\mathbf{\Phi}$ to the clients. Clients use this matrix to encode their samples $\mathbf{X}$ into hypervectors $\mathbf{H}$ (Fig. 2 step 1).

2. In each training epoch, clients compute local class hypervector updates using Eq. 2 based on their current $\mathbf{H}, \mathbf{C}$ (Fig. 2 step 2). The updated local class HVs are sent to the server. The server averages these to form global class HVs, which are then broadcasted back to all clients.

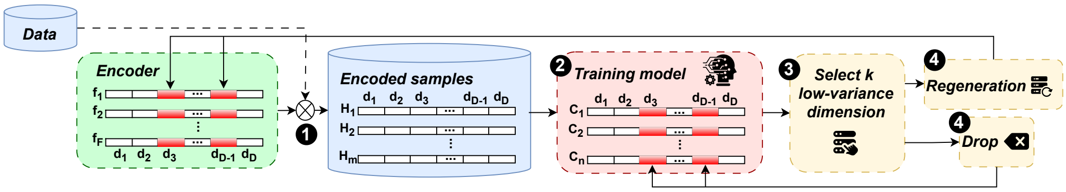


Figure 2: HDC training with dimension regeneration. 1) Clients encode their samples through the encoding matrix; 2) The encoded samples are used to update the class HVs; 3) The dimensions of the class HVs with the lowest contribution to the classification task are identified; 4) The identified dimensions are set to zero in the class HVs, and the corresponding dimensions in the encoding matrix are regenerated;

3. At expected regeneration epochs, clients locally compute the dimension contribution scores based on Eq. (3) using local data and current global model (Fig. 2 step 3). These scores are sent to the server.
4. The server sums these scores across all clients, determines the global set $\mathcal{D}'$ of k dimensions to regenerate, generates new random vectors for these dimensions in $\mathbf{\Phi}$ (Fig. 2 step 4), and distributes the updated encoder and the indices $\mathcal{D}'$ to the clients. Clients update their local $\mathbf{\Phi}$, re-encode affected data if necessary, and update their class hypervectors $\mathbf{C}$.

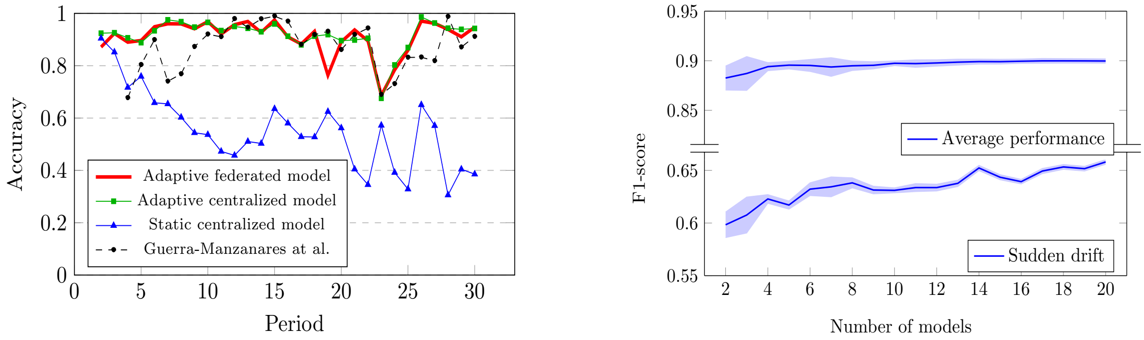
During inference, predictions on a query sample H_q are made by combining the outputs of the individual models $m \in \{1, \ldots, M\}$ through majority voting. This architecture intrinsically handles resource heterogeneity. Clients can adaptively select the number of ensemble models they participate in training based on their local computational capabilities. Similarly, during inference, clients can choose to query all M models or a subset thereof, balancing prediction latency and accuracy according to their constraints. This flexibility makes the system practical for deployment across heterogeneous clients.

4. Experimental evaluation

Experiments were conducted on the Kronodroid dataset [20], containing 78,137 samples (41,382 malware, 36,755 legitimate) collected over 8 years. The dataset was divided into three-month chunks based on APK modification dates, consistent with [21]. The first chunk was used for initial training and excluded from evaluation to avoid temporal bias. A prequential evaluation [22] with a test-then-train scheme was applied to subsequent chunks, ensuring models were evaluated on unseen data to prevent temporal snooping [23]. For each chunk, the model was trained for 10 epochs, with a regeneration frequency of 2 epochs and a regeneration rate of 0.04, as in [17]. The overall dimensionality of the model was set to 2000, which was subsequently partitioned across the ensemble models. The experiments were evaluated through accuracy and F1-score. The experiments were carried out with 10 federated clients.

First, an analysis was performed to assess the impact of concept drift, and determine whether the federated approach can be an effective alternative to centralized training. The results of this analysis are reported in Fig. 3a. A static model trained exclusively on the first three months of data shows a steep accuracy degradation of almost 20% after just 6 months, and the accuracy eventually drops to below random guessing. Instead, by retraining every three months, both centralized and federated approaches maintain high accuracy over time and recover quickly from sudden drifts. Although centralized training yields slightly more accurate models, with a change in average performance from 0.92 to 0.91 between the centralized and federated models this difference is not statistically significant with $p > 0.05$, while the difference between the static model and the adaptive model is significant ($p < 2 \cdot 10^{-12}$). Moreover, the proposed architecture slightly outperforms existing approaches [21] that use traditional centralized tree-based ensembles on this dataset.

In order to investigate the impact of resource limits, the inference phase was evaluated with different ensemble sizes as shown in Fig. 3b. On average, performance remains comparable regardless of the number of models. However, in the presence of sudden drift, larger ensembles show greater resilience capacity, mitigating performance degradation more effectively. This result opens up the possibility of using a smaller number of models for inference, especially in scenarios with limited computational



(a) Differences between a static and adaptive centralized/federated model, showing the presence of drift over time. The federated and adaptive models show comparable performance.

(b) System performance varying ensemble size during inference. On average small subsets of the ensemble achieve satisfactory performance. However, when sudden drift occurs, larger ensembles are more robust.

Figure 3: Experimental results

resources, in conjunction with a drift detection mechanism that can trigger the use of a larger ensemble.

5. Conclusions and future works

Malware detection is a critical aspect of mobile device protection. The constant evolution of threats requires continuous updates to malware detection systems. However, the collection and sharing of the data required for training can raise significant privacy issues. For this reason, decentralized approaches capable of analyzing and updating models directly on the devices are preferable. The proposed architecture introduces a novel approach for using Hyperdimensional Computing-based model ensembles in a federated context, enabling collaborative training between devices while maintaining the privacy of local data. Furthermore, the proposed architecture is particularly suitable for real-world applications as it effectively addresses the problem of client heterogeneity, allowing the participation of devices with limited computational resources without the need to use the entire model ensemble for inference. Future works will investigate dynamic ensemble weighting and selection strategies [24, 25], and a drift detection mechanism to further improve the efficiency of the proposed architecture.

Acknowledgements This work was partially supported by the AMELIS project, within the project FAIR (PE0000013), and by the ADELE project, within the project SERICS (PE00000014), both under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] K. Allix, T. F. Bissyandé, J. Klein, Y. Le Traon, Are your training datasets yet relevant? An investigation into the importance of timeline in machine learning-based malware detection, in: Int. Symp. on Engineering Secure Software and Systems, 2015, pp. 51–67.

[2] H. Razeghi Borojerdi, M. Abadi, Malhunter: Automatic generation of multiple behavioral signatures for polymorphic malware detection, in: ICCKE 2013, 2013, pp. 430–436. doi:10.1109/ICCKE.2013.6682867.

[3] Z. Liu, R. Wang, N. Japkowicz, D. Tang, W. Zhang, J. Zhao, Research on unsupervised feature learning for android malware detection based on restricted boltzmann machines, Future Generation Computer Systems 120 (2021) 91–108.

[4] A. Augello, A. De Paola, G. Lo Re, Hybrid multilevel detection of mobile devices malware under concept drift, J. of Network and Systems Management 33 (2025) 36. doi:10.1007/s10922-025-09906-3.

[5] N. Lu, G. Zhang, J. Lu, Concept drift detection via competence models, Artificial Intelligence 209 (2014) 11–28.

[6] V. Agate, A. De Paola, S. Drago, P. Ferraro, G. Lo Re, Enhancing IoT network security with concept drift-aware unsupervised threat detection, in: 2024 IEEE Symp. on Computers and Communications (ISCC), 2024, pp. 1–6.

[7] V. Agate, S. Drago, P. Ferraro, G. Lo Re, Anomaly detection for reoccurring concept drift in smart environments, in: 18th Int. Conf. on Mobility, Sensing and Networking, 2022, pp. 113–120.

[8] A. De Paola, S. Drago, P. Ferraro, G. Lo Re, Detecting zero-day attacks under concept drift: An online unsupervised threat detection system, in: CEUR-WS Proc. ITASEC, volume 3731, 2024.

[9] F. Camarda, A. De Paola, S. Drago, P. Ferraro, G. Lo Re, Managing concept drift in online intrusion detection systems with active learning, in: CEUR-WS Proc. Joint National Conf. on Cybersecurity, ITASEC & SERICS, volume 2025, 2025.

[10] A. Augello, A. De Paola, G. Lo Re, M2FD: Mobile malware federated detection under concept drift, Computers & Security 152 (2025) 104361. doi:10.1016/j.cose.2025.104361.

[11] B. Ghimire, D. B. Rawat, Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things, IEEE Internet of Things J. 9 (2022) 8229–8249.

[12] F. Concone, C. Ferdico, G. Lo Re, M. Morana, A federated learning approach for distributed human activity recognition, in: 2022 IEEE Int. Conf. on Smart Computing (SMARTCOMP), 2022, pp. 269–274.

[13] C. B. Mawuli, L. Che, J. Kumar, S. U. Din, Z. Qin, Q. Yang, J. Shao, Fedstream: Prototype-based federated learning on distributed concept-drifting data streams, IEEE Trans. on Systems, Man, and Cybernetics: Systems 53 (2023) 7112–7124.

[14] E. Jothimurugesan, K. Hsieh, J. Wang, G. Joshi, P. B. Gibbons, Federated learning under distributed concept drift, in: Int. Conf. on Artificial Intelligence and Statistics, 2023, pp. 5834–5853.

[15] A. Augello, A. De Paola, G. Lo Re, A. Menager, HDDroid: Federated hyperdimensional computing for mobile malware detection, in: CEUR Workshop Proceedings, 2025.

[16] A. Thomas, S. Dasgupta, T. Rosing, A theoretical perspective on hyperdimensional computing, J. of Artificial Intelligence Research 72 (2021) 215–249.

[17] J. Wang, S. Huang, M. Imani, DistHD: A learner-aware dynamic encoding method for hyperdimensional classification, in: 2023 60th ACM/IEEE Design Automation Conf., 2023.

[18] Y. Tian, R. Chandrasekaran, K. Ergun, X. Yu, T. Rosing, Federated hyperdimensional computing: Comprehensive analysis and robust communication, ACM Trans. on Internet of Things (2025).

[19] R. Wang, D. Ma, X. Jiao, Enhdc: Ensemble learning for brain-inspired hyperdimensional computing, IEEE Embedded Systems Letters 15 (2022) 37–40.

[20] A. Guerra-Manzanares, H. Bahsi, S. Nõmm, Kronodroid: Time-based hybrid-featured dataset for effective android malware detection and characterization, Computers & Security 110 (2021) 102399.

[21] A. Guerra-Manzanares, M. Luckner, H. Bahsi, Android malware concept drift using system calls: detection, characterization and challenges, Expert Systems with Applications 206 (2022) 117200.

[22] J. Gama, R. Sebastiao, P. P. Rodrigues, Issues in evaluation of stream learning algorithms, in: Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining, 2009, pp. 329–338.

[23] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, L. Cavallaro, TESSERACT: Eliminating experimental bias in malware classification across space and time, in: 28th USENIX security symposium (USENIX Security 19), 2019, pp. 729–746.

[24] V. Agate, F. Concone, A. De Paola, P. Ferraro, S. Gaglio, G. Lo Re, M. Morana, Adaptive ensemble learning for intrusion detection systems, in: CEUR Workshop Proceedings - 4th National Conf. on Artificial Intelligence, Ital-IA 2024, volume 3762, 2024, pp. 118–123.

[25] V. Agate, F. M. D'anna, A. De Paola, P. Ferraro, G. Lo Re, M. Morana, A behavior-based intrusion detection system using ensemble learning techniques., in: ITASEC, 2022, pp. 207–218.