INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE, CYBERSECURITY AND EMERGING TECHNOLOGIES (IJAICET)

IJAICET, Vol. 1, Issue 1, July - September, 2026

Random Forest over Semantic PDF Object Graphs for Robust Malware Detection

Mr. Mahesh Kumar Bagwani and Dr. Ganesh Gupta

Abstract-Portable Document Format (PDF) malware remains a persistent cybersecurity threat because the format supports indirect objects, compressed streams, embedded JavaScript, actions, annotations, file attachments, and complex cross-reference structures. This paper presents an expanded study of a classical Random Forest classifier applied to semantic PDF Object Reference Graphs derived from the public PDFObj2Vec dataset. Instead of training a graph neural network, each PDF is represented by BERT-65k object embeddings pooled at graph level and augmented with structural graph statistics. The resulting 4107-dimensional vector is used for binary malware classification. The work formalises the graph-to-vector transformation, impurity-based tree induction, ensemble voting, class-probability estimation, and evaluation metrics. Experiments on the PDFObj2Vec train, baseline test, and extended evaluation splits show that the Random Forest achieves 99.97% baseline accuracy and 96.10% extended accuracy, remaining close to the reported BERT-65k+GIN result while offering simpler implementation, lower deployment complexity, and feature-group interpretability. The study also discusses operational deployment, false-positive/false-negative trade-offs, adversarial validity, and concept-drift risks for real-world PDF malware screening.

Impact Statement-This study provides a reproducible and interpretable baseline for AI-based PDF malware detection. The method is useful for security teams that require accurate static screening without the operational cost of graph neural inference. It also links PDF malware classification with the author's previous work on cloud security, DDoS defence, web deployment, microservices, QR-code malware risk, and cloud machine-learning systems.

Index Terms-PDF malware detection, Random Forest, PDFObj2Vec, PDF object graph, BERT embeddings, graph representation, static malware analysis, cloud security, adversarial machine learning, cybersecurity.

I. INTRODUCTION

PDF files are exchanged as invoices, contracts, academic articles, scanned records, forms, reports, and official correspondence. Their ubiquity makes them attractive to attackers because a malicious document can be delivered through email, messaging systems, cloud storage, or web portals while still appearing to be a normal business document. The PDF specification permits nested objects, streams, actions, annotations, JavaScript, embedded files, references, encryption, and incremental updates. These features are useful for legitimate documents, but they also create a large attack surface for concealment, obfuscation, and parser confusion.

Traditional static detectors often count suspicious keywords, JavaScript objects, launch actions, embedded files, object types, or metadata fields. These signals remain useful but can be brittle when attackers add benign-looking objects, rename fields, compress streams, or manipulate document structure without changing the malicious behaviour. Dynamic analysis can observe execution but is expensive, slower, and sometimes ineffective when the malware checks its environment. A practical detector should therefore combine semantic evidence from PDF objects with structural evidence from object references.

Recent learning-based approaches move beyond handcrafted indicators by representing PDF documents as graphs. In the PDFObj2Vec setting, every PDF is parsed into an Object Reference Graph (ORG), where nodes represent PDF objects and edges represent references. Node embeddings produced by a BERT-65k representation model encode semantic information at the object level. The base PDFObj2Vec paper demonstrates that a BERT-65k embedding combined with a Graph Isomorphism Network (GIN) achieves strong accuracy. This paper investigates whether a simpler Random Forest can use the same semantic graph evidence after fixed-length vectorisation.

This direction is important for operational cybersecurity. Classical ensembles can be easier to train, deploy, monitor, explain, and reproduce than graph neural networks. They are suitable for cloud-native services, malware triage systems, and security gateways where latency, model transparency, and maintainability matter. The author's earlier work on DDoS defence in cloud environments [29], AWS Amplify deployment [30], GrapesJS educational platforms [31], cloud machine-learning services [32], REST and GraphQL microservice performance [33], and microservice architecture comparison [34] motivates a deployment-aware view of the PDF malware detector rather than a purely benchmark-oriented view.

A. MAIN CONTRIBUTIONS

A detailed graph-to-vector formulation that converts variable-size PDF Object Reference Graphs into fixed-length 4107-dimensional vectors.

A Random Forest classifier using BERT-65k semantic object embeddings and eleven structural graph statistics.

Corrected mathematical expressions for semantic pooling, graph statistics, Gini impurity, ensemble voting, posterior probability estimation, and evaluation metrics.

An expanded experimental discussion covering baseline, extended, and operational-security interpretations of the results.

A related-work extension that includes the author's cybersecurity, cloud, QR-code malware, and microservice publications as contextual foundations for deployment and threat modelling.

II. RELATED WORK

A. STATIC PDF MALWARE DETECTION

Static PDF malware detection studies show that document structure, metadata, object hierarchy, and embedded scripting features provide strong discriminative evidence. Laskov and Srndic [2], Smutz and Stavrou [3], Srndic and Laskov [4], and Maiorca et al. [5], [6] established that static PDF features can

detect many malicious documents efficiently. However, evasion studies [7], [8] show that learning-based PDF classifiers can be manipulated when attackers alter non-functional content, add benign objects, or exploit features that are highly correlated with the training corpus but not causally tied to malicious behaviour.

B. Representation Learning for Security

Representation learning has improved malware analysis by converting code tokens, assembly instructions, program graphs, and document structures into dense vectors. BERT [16], graph neural networks [17], binary embedding methods [13]-[15], and graph anomaly detection [24] provide the foundation for semantic and structural modelling. PDFObj2Vec extends this idea to PDF objects by learning object-level representations and using graph neural classification. The present study keeps the same semantic representation but replaces graph neural inference with a Random Forest over pooled graph-level features.

C. Cloud Security, Deployment, and QR-Code Threat Context

The proposed detector is not isolated from broader cybersecurity infrastructure. Bagwani et al. [29] studied real-time signature-based DDoS detection and prevention in cloud environments, emphasising rapid detection and operational response. Bagwani [30] discussed deployment of web applications on AWS Amplify, while Bagwani and Tiwari [31], [37] examined GrapesJS integration with AWS for educational platforms. These works provide a deployment background for converting a research model into a service-facing security component.

Cloud machine-learning deployment is also relevant. Bagwani and Tiwari [32] studied cloud machine-learning services for face-detection performance, indicating the importance of managed inference, latency, and scalability. Bagwani [33], Bagwani and Shrivastava [34], and Bagwani's cloud-native GraphQL and Docker work [38] discuss microservice performance and maintainability. These concerns apply directly to PDF malware screening, where a detector may be packaged as a microservice behind an upload gateway or enterprise mail pipeline.

QR-code malware research by Bagwani et al. [35] and Bagwani and Tiwari [36] is conceptually related because both QR-code threats and PDF malware exploit trusted document-like carriers. The shared problem is that a benign-looking object can encode a malicious destination, payload, or trigger. This paper therefore treats PDF malware detection as part of a wider family of document-mediated and user-facing attack detection problems.

III. PROBLEM FORMULATION

Let P denote the set of PDF files and let p in P denote a single PDF document. Each document is transformed into an Object Reference Graph in which PDF objects become nodes and object references become directed edges.

$$G_p = (V_p, E_p, X_p) \quad (1)$$

Here, V_p is the node set, E_p is the directed edge set, and X_p is the node-feature matrix.

$$X_p \in \mathbb{R}^{npxd}, \quad n_p = |V_p|, \quad d = 1024 \quad (2)$$

The binary label is defined as follows, where zero denotes benign and one denotes malicious.

$$y_p \in \{0, 1\} \quad (3)$$

A graph neural classifier directly estimates f_theta(G_p). A Random Forest requires fixed-length input, so a deterministic mapping phi(.) is used to convert the graph into a vector.

$$\varphi : (V_p, E_p, X_p) \rightarrow \mathbb{R}^{4107} \quad (4)$$

$$h(\varphi(G_p)) = \hat{y}_p, \quad \hat{y}_p \in \{0, 1\} \quad (5)$$

The learning objective is to estimate h(.) such that empirical classification error on unseen PDFs is minimised while preserving operational interpretability.

IV. PROPOSED METHODOLOGY

A. Semantic Pooling of Object Embeddings

For PDF p, the node embeddings are represented as x_1, x_2, ..., x_np, where each x_i is a 1024-dimensional BERT-65k object embedding. Four element-wise pooling operators are used. Mean pooling captures the average semantic behaviour of all objects.

$$\mu_p = (1 / n_p) \Sigma_{i=1}^{np} x_i \quad (6)$$

Maximum and minimum pooling preserve extreme activations that may correspond to rare suspicious objects, unusually benign object clusters, or obfuscated embedded content.

$$x_p^{max}[j] = \max_{1 \le i \le np} x_i[j] \quad (7)$$

$$x_p^{min}[j] = \min_{1 \le i \le np} x_i[j] \quad (8)$$

Standard-deviation pooling captures heterogeneity across PDF objects.

$$\sigma_p[j] = \sqrt{((1 / n_p) \Sigma_{i=1}^{np} (x_i[j] - \mu_p[j])^2)} \quad (9)$$

The semantic graph-level vector is the concatenation of the four pooled blocks.

$$s_p = [\mu_p \,||\, x_p^{max} \,||\, x_p^{min} \,||\, \sigma_p] \in \mathbb{R}^{4096} \quad (10)$$

B. Structural Graph Statistics

Semantic pooling compresses object content but loses explicit information about graph shape. Therefore, eleven structural statistics are added: node count, edge count, density, mean in-degree, maximum in-degree, mean out-degree, maximum out-degree, mean total degree, maximum total degree, isolated-node ratio, and leaf-node ratio.

$$\rho_p = |E_p| / \max(1, |V_p|(|V_p|-1)) \quad (11)$$

$$d_{out}(v_i) = |\{(v_i, v_j) \in E_p\}|, \quad d_{in}(v_i) = |\{(v_j, v_i) \in E_p\}| \quad (12)$$

$$d(v_i) = d_{in}(v_i) + d_{out}(v_i) \quad (13)$$

$$g_p \in \mathbb{R}^{11}, \quad \varphi(G_p) = [s_p \,||\, g_p] \in \mathbb{R}^{4107} \quad (14)$$

C. Random Forest Classifier

A Random Forest consists of B decision trees trained on bootstrap samples. At each split, a random subset of features is considered. The split criterion minimises class impurity. For a node sample S, the Gini impurity is:

$$Gini(S) = 1 - \Sigma c \in \{0,1\} \, p^{c2} \quad (15)$$

The prediction is obtained by majority voting across trees.

$$\hat{y}_p = \text{argmax}_{c \in \{0,1\}} \Sigma^{b}_{=1}{}^{B} I(T_\beta(\varphi(G_p)) = c) \quad (16)$$

The malicious-class probability is estimated as the fraction of trees voting for the malicious class.

$$\Pr(y_p=1 \mid G_p) = (1 / B) \Sigma^{b}_{=1}{}^{B} I(T_\beta(\varphi(G_p)) = 1) \quad (17)$$

TABLE I. PROPOSED SEMANTIC GRAPH-TO-VECTOR RANDOM FOREST WORKFLOW.

Step	Operation	Output
1	Parse PDF and construct Object Reference Graph	G_p
2	Load BERT-65k embedding for each PDF object	X_p
3	Apply mean, max, min, and standard-deviation pooling	s_p
4	Compute graph statistics from object-reference edges	g_p
5	Concatenate semantic and structural feature blocks	phi(G_p)
6	Train Random Forest and evaluate on held-out splits	metrics

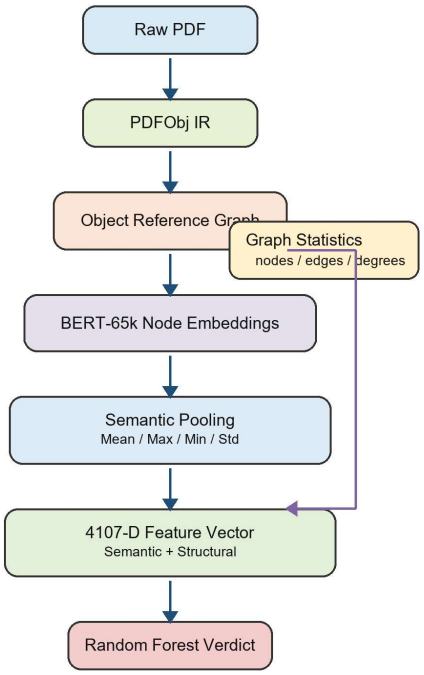

Fig. 1. Research pipeline for semantic PDF object-graph embedding and Random Forest classification.

V. EXPERIMENTAL DESIGN

A. DATASET

The experiment uses the public PDFObj2Vec evaluation dataset. The dataset provides graph files, BERT-65k object embeddings, and binary labels for benign and malicious PDF documents. Three splits are used: a training split for model fitting, a baseline test split for in-distribution evaluation, and an extended split for harder evaluation under broader document diversity.

TABLE II. DATASET SPLITS USED IN THIS STUDY.

Split	Samples	Benign	Malicious	Purpose
Train	13,190	-	-	Model fitting
Baseline test	6,133	2,698	3,435	In-distribution evaluation
Extended test	45,947	24,476	21,471	Broader robustness evaluation

B. FEATURE-DIMENSION SUMMARY

TABLE III. FEATURE BLOCKS IN THE PROPOSED GRAPH-LEVEL VECTOR.

Feature block	Dimension	Description
Mean pooled BERT-65k embedding	1024	Average semantic signal across PDF objects
Maximum pooled embedding	1024	Strongest object-level activation per dimension
Minimum pooled embedding	1024	Lowest object-level activation per dimension
Standard-deviation pooled embedding	1024	Object-level semantic variability
Structural statistics	11	Counts, degree statistics, density, isolation, and leaf ratios
Total	4107	Complete Random Forest input vector

C. EVALUATION METRICS

Let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives. The evaluation metrics are defined as follows:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (18)$$

$$\text{Precision} = TP / (TP + FP) \quad (19)$$

$$\text{Recall} = TPR = TP / (TP + FN) \quad (20)$$

$$TNR = TN / (TN + FP) \quad (21)$$

$$F1 = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall}) \quad (22)$$

VI. RESULTS AND ANALYSIS

A. Classification Performance

The Random Forest achieves perfect training performance, near-perfect baseline-test performance, and strong extended-set performance. The larger drop on the extended split is expected because it contains a broader set of benign and malicious documents and therefore tests robustness beyond the nearest in-distribution setting.

TABLE IV. Random Forest performance on PDFObj2Vec BERT-65k graph embeddings.

Split	Accuracy	Precision	Recall/TPR	TNR	F1-score
Train	100.00	100.00	100.00	100.00	100.00
Baseline test	99.97	100.00	99.94	100.00	99.97
Extended test	96.10	94.29	97.55	94.82	95.90

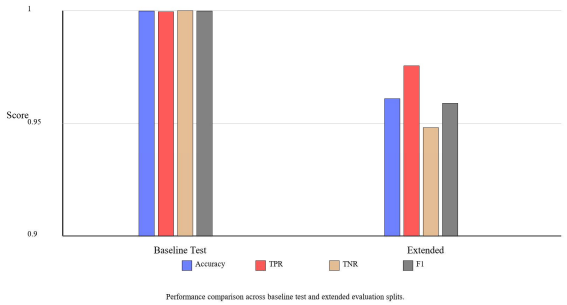

Fig. 2. Performance comparison across baseline and extended evaluation splits.

B. Confusion-Matrix Interpretation

The extended split contains 23,208 true negatives, 1,268 false positives, 525 false negatives, and 20,946 true positives. The model therefore favours high malicious recall, which is valuable when missed malware is more costly than manual review of suspicious benign documents. The 1,268 false positives still matter in production because excessive alert volume can reduce analyst trust. Threshold calibration and a secondary review model may reduce this cost.

TABLE V. Confusion matrices for all evaluation splits.

Split	TN	FP	FN	TP
Train	-	-	-	-
Baseline test	2,698	0	2	3,433
Extended test	23,208	1,268	525	20,946

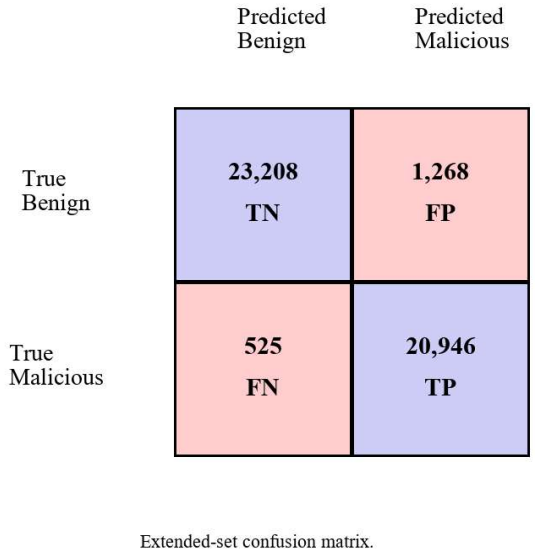

FIG. 3. Extended-set confusion matrix for the Random Forest classifier.

C. Comparison with Graph Neural Classification

The base PDFObj2Vec paper reports 99.93% baseline accuracy and 96.62% extended accuracy for the BERT-65k+GIN classifier. The Random Forest reaches 99.97% baseline accuracy and 96.10% extended accuracy. The extended-set gap is only 0.52 percentage points, indicating that the pooled semantic object representation already contains substantial discriminative information.

TABLE VI. Comparison with reported BERT-65k+GIN performance.

Model	Baseline accuracy	Extended accuracy	Operational interpretation
BERT-65k + GIN	99.93	96.62	Higher graph-structure modelling capacity
BERT-65k pooled vector + Random Forest	99.97	96.10	Simpler training, easier deployment, interpretable feature groups

VII. DISCUSSION

The results suggest that much of the discriminative evidence is encoded in the BERT-65k object embeddings before graph neural message passing is applied. Mean pooling captures the general semantic tendency of the document; maximum and minimum pooling capture rare object-level extremes; and standard-deviation pooling captures semantic heterogeneity. The structural feature block then restores coarse graph evidence that pooling alone would remove.

From a deployment perspective, the Random Forest is attractive because it requires only vector input at inference time. After PDF parsing and embedding generation, prediction can be served as a standard microservice. This aligns with the author's prior work on AWS deployment [30], cloud ML services [32], REST/GraphQL microservice comparison [33], microservice architecture analysis [34], and Docker-based cloud-native application design [38]. A security gateway can expose the detector as an internal API and log probability scores, feature-block summaries, and decision outcomes for audit.

The model also supports feature-group analysis. Although a single BERT embedding dimension is not directly human-readable, importances can be aggregated by mean, maximum, minimum, standard deviation, and structural blocks. This can tell analysts whether a decision was driven mainly by common semantic content, rare suspicious object activations, abnormal variability, or graph topology. Such grouping is useful for security operations where explainability affects analyst trust.

The security trade-off should be configured by risk. In an email gateway, a higher recall threshold may be preferred to reduce missed malware, even at the cost of more false positives. In a public upload portal, false positives may cause user friction, so the threshold may need calibration. In a forensic triage setting, probability ranking may be more useful than a hard benign/malicious label.

VIII. THREATS TO VALIDITY

Internal validity is limited by reliance on precomputed embeddings. Errors in PDF parsing, object extraction, graph construction, or BERT-65k embedding generation propagate into the Random Forest. The experiment therefore evaluates the classifier and graph-level pooling strategy rather than the entire PDF analysis pipeline from raw bytes to final verdict.

External validity depends on dataset coverage. Real-world document streams may contain encrypted PDFs, password-protected files, scanned-only documents, malformed objects, language-specific templates, digitally signed files, and enterprise-generated forms. The extended split partly addresses distribution diversity, but deployment should include continuous monitoring.

Adversarial validity remains open. The present experiment does not evaluate adaptive attacks such as object insertion, graph perturbation, stream manipulation, reverse mimicry, or targeted attacks against tree ensembles. Prior work on evasion and hardening [7], [8], [21], [22] indicates that this must be tested before high-stakes deployment.

IX. FUTURE WORK

Evaluate the Random Forest under adversarial PDF manipulation, reverse mimicry, and graph perturbation.

Compare Random Forest with XGBoost, LightGBM, support vector machines, logistic regression, and calibrated neural models using the same feature vector.

Develop feature-group explanations that aggregate importance by semantic pooling block and structural graph statistics.

Evaluate temporal concept drift using timestamp-based splits and malware-family-aware validation.

Deploy the detector as a cloud-native microservice and measure latency, throughput, cost, and alert-review burden.

Extend the method to document-mediated threats such as malicious QR-code payloads, embedded URLs, and multimodal phishing documents.

X. CONCLUSION

This paper presented an expanded study of Random Forest classification over semantic PDF Object Reference Graph embeddings. By pooling BERT-65k node representations and adding structural graph statistics, the method converts variable-size PDF graphs into 4107-dimensional vectors suitable for classical ensemble learning. The resulting classifier achieves 99.97% baseline accuracy and 96.10% extended accuracy on the public PDFObj2Vec evaluation setting, remaining close to the reported BERT-65k+GIN model while offering simpler deployment and practical interpretability. The study shows that classical ensemble learning remains a strong and operationally meaningful baseline for PDF malware detection when semantic graph embeddings are available.

XI. ACKNOWLEDGMENT

The authors acknowledge Amity University Madhya Pradesh for academic support during this research work.

REFERENCES

[1] CISA, "Phishing guidance and cyber hygiene resources," Cybersecurity and Infrastructure Security Agency, 2022.

[2] P. Laskov and N. Srndic, "Static detection of malicious JavaScript-bearing PDF documents," ACSAC, 2011.

[3] C. Smutz and A. Stavrou, "Malicious PDF detection using metadata and structural features," ACSAC, 2012.

[4] N. Srndic and P. Laskov, "Hidost: a static machine-learning-based detector of malicious files," EURASIP Journal on Information Security, 2016.

[5] D. Maiorca, G. Giacinto, and I. Corona, "A pattern recognition system for malicious PDF files detection," MLDM, 2012.

[6] D. Maiorca et al., "Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious PDF files detection," ASIACCS, 2013.

[7] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," NDSS, 2016.

[8] N. Srndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," IEEE Symposium on Security and Privacy, 2014.

[9] L. Tong et al., "Improving robustness of ML classifiers against realizable evasion attacks using conserved features," USENIX Security, 2019.

[10] Y. Chen et al., "Robust PDF malware detection with adversarial training and static features," security literature, 2020.

[11] A. Jordaney et al., "Transcend: Detecting concept drift in malware classification models," USENIX Security, 2017.
[12] F. Barbero et al., "Transcending Transcend: Revisiting malware concept drift detection," security literature, 2022.
[13] S. Chua et al., "Neural nets can learn function type signatures from binaries," USENIX Security, 2017.
[14] S. Ding et al., "Asm2Vec: Boosting static representation robustness for binary clone search," IEEE S&P, 2019.
[15] X. Li et al., "PalmTree: Learning an assembly language model for instruction embedding," CCS, 2021.
[16] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," NAACL, 2019.
[17] K. Xu et al., "How powerful are graph neural networks?" ICLR, 2019.
[18] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.
[19] Adobe Systems, "PDF Reference and ISO 32000-1 portable document format specification," 2008.
[20] PDF Association, "Stressful PDF corpus and PDF parser robustness resources," 2020.
[21] A. Kantchelian et al., "Evasion and hardening of tree ensemble classifiers," ICML, 2016.
[22] A. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," Pattern Recognition, 2018.
[23] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," KDD, 2016.
[24] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description," Data Mining and Knowledge Discovery, 2015.
[25] M. Newman, Networks, Oxford University Press, 2018.
[26] DGL Team, "Deep Graph Library," software documentation, 2024.
[27] A. Anantharaman et al., "PolyDoc: A corpus for PDF parser differential analysis," 2023.
[28] QPDF Project, "QPDF manual and PDF transformation tools," 2024.
[29] M. K. Bagwani, V. K. Tiwari, A. Gangwar, and K. Vishwakarma, "Real-time signature-based detection and prevention of DDOS attacks in cloud environments," International Journal of Science and Research Archive, vol. 12, no. 2, pp. 2929-2935, 2024.
[30] M. Bagwani, "Deploying a web application on AWS Amplify: A comprehensive guide," International Journal of Progressive Research in Engineering Management and Science, 2024.
[31] A. M. K. Bagwani and V. K. Tiwari, "Implementing GrapesJS in educational platforms for web development training on AWS," International Journal of Scientific Research in Multidisciplinary Studies, vol. 10, 2024.
[32] A. M. K. Bagwani and V. K. Tiwari, "Optimizing face detection performance with cloud machine learning services," Journal of Engineering and Technology Management, vol. 73, pp. 1167-1180, 2024.
[33] P. G. K. S. Mahesh Kumar Bagwani, "Performance comparison of REST API and GraphQL in a micro services architecture," International Conference on Data Science, Artificial Intelligence and Advanced Computing, 2024.
[34] M. K. Bagwani and G. K. Shrivastava, "Comparative analysis of microservices architectures: Evaluating performance, scalability, and maintenance," International Journal on Advances in Engineering Technology and Science, vol. 5, 2023.
[35] M. K. Bagwani, S. Dwivedi, V. Kumar, and P. Koshti, "Transmitting malware through QR codes: Risk analysis and a hybrid detection method," International Journal for Multidisciplinary Research, vol. 8, no. 3, pp. 1-25, 2026.
[36] A. M. K. Bagwani and V. K. Tiwari, "Preventing malware spread through QR codes: A detection and analysis approach," Journal of Engineering and Technology Management, vol. 73, pp. 1260-1268, 2024.
[37] A. M. K. Bagwani, V. K. Tiwari, and N. Singh, "Integrating GrapesJS with AWS: Building an educational platform for web development training," An Overview of Literature, Language and Education Research, vol. 10, pp. 106-124, 2025.
[38] M. Bagwani, "Building a cloud-native microservices based web application with GraphQL and Docker," School of Advanced Computing, Sanjeev Agrawal Global Educational University, 2024.