

Semantic PDF Object Embedding Pooling with Random Forest for Malware Detection

Nihal Thakur¹, Prajwal Raghuvanshi², Pushkar Kushwaha³, Niranjana Saxena⁴, and Manas Khutke⁵

School of Computer Technology, Sanjeev Agrawal Global Educational University, Bhopal*

Abstract: This study evaluates a Random Forest on precomputed BERT-65k PDF object embeddings from the PDFObj2Vec evaluation dataset by combining four pooling operations with eleven graph statistics to form 4,107 features per document; the classifier correctly labels 6,131 of 6,133 baseline test PDFs for 99.97% accuracy and 44,154 of 45,947 extended set PDFs for 96.10% accuracy while producing 1,268 false positives and 525 false negatives on the latter set; the published BERT-65k+GIN result is 96.62% extended accuracy, although this aggregate comparison does not establish statistical equivalence or a computational advantage because adversarial robustness, split independence, calibration and end to end latency remain untested.

Impact Statement: Object embedding pooling supports classification without graph neural message passing at the final decision stage, yet the measured extended set false positive rate of 5.18% and false negative rate of 2.45% together with unmeasured parsing and embedding costs mean that independent temporal testing, adversarial evaluation, calibration and operational measurement are needed before deployment.

Index Terms: BERT embeddings, graph pooling, malware detection, object reference graph, PDF, Random Forest.

I. INTRODUCTION

PDF files are widely exchanged as business records and academic documents, making them attractive to attackers who can deliver malicious content through familiar channels while the format's nested objects, streams, actions, embedded files and incremental updates create opportunities for concealment and parser confusion.

Traditional static detectors use keywords alongside object and metadata features, but these signals can be weakened by apparently benign additions or structural manipulation, while dynamic analysis is slower and can be evaded through environment checks; consequently, a detector should consider both object semantics and reference structure.

The distinction between a suspicious object and a malicious document matters because ordinary files may contain active features for legitimate purposes, so a document classifier must combine evidence across many objects rather than treat one keyword or one structural feature as a definitive verdict. PDFObj2Vec represents PDFs as object reference graphs and learns object embeddings for graph classification [29], whereas this study evaluates a downstream Random Forest on the released embeddings [30] without reassessing the original parser or embedding model.

This study asks whether pooled semantic embeddings retain enough discriminative information for tabular classification while treating simpler training, computational savings and interpretability as hypotheses that require separate measurement.

A. Main Contributions

A fixed length representation combining four 1,024-dimensional embedding summaries with eleven reported graph statistics.

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

An evaluation of the reported train, baseline test, and extended set confusion matrices, including security relevant error rates. A clearly bounded comparison with published GIN accuracy, without claiming paired significance or adversarial robustness.

A reproducibility and validation agenda covering dataset identity, split overlap, ablations, calibration, and runtime.

II. RELATED WORK

A. Static PDF Malware Detection

Static PDF studies show that document structure and embedded content features can support detection [2]-[6], although evasion research shows that attackers may manipulate classifiers by adding nonfunctional objects or exploiting correlations that do not reflect malicious behaviour [7], [8].

B. Representation Learning for Security

A learned object representation can capture regularities that are difficult to express through fixed keyword lists, yet the classifier still depends on how the original PDF was parsed and on whether the training collection reflects the documents encountered during evaluation.

Representation learning converts code and document structure into dense vectors through methods such as BERT [16] and graph neural networks [17], while PDFObj2Vec applies this approach to PDF objects [29] and the present study replaces its graph neural classification stage with a Random Forest over pooled graph features.

C. Deployment Scope and Operational Evaluation

Deployment depends on parsing, embedding, preprocessing and alert handling as well as the classifier itself, but the

reported experiment does not measure the latency, failure modes or cost of those upstream and downstream stages.

Because the experiment uses precomputed graph embeddings, its findings concern downstream classification rather than a complete PDF screening gateway.

An operational evaluation should measure precision under realistic prevalence and analyst workload while separately testing encrypted or malformed files and adversarial PDF edits that are absent from the reported experiment.

III. PROBLEM FORMULATION

Let P denote the set of PDFs and p denote one document whose Object Reference Graph represents PDF objects as nodes and object references as 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.

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

The binary label is zero for benign documents and one for malicious documents.

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

Unlike a graph neural classifier, a Random Forest requires fixed length input and therefore uses the deterministic mapping ϕ to convert each graph into a vector.

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

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

The supervised objective is to predict labels for unseen graphs while holding feature extraction and data partitions fixed for a fair classifier comparison.

IV. PROPOSED METHODOLOGY

A. Semantic Pooling of Object Embeddings

Pooling is necessary because PDFs contain different numbers of objects while the Random Forest expects a fixed number of inputs, and the four selected summaries retain complementary information about central tendency, extreme activations and variation without preserving the full ordering of nodes.

For document p , each of its n_p nodes has a 1,024-dimensional BERT-65k embedding and mean pooling summarizes the average object representation before three further elementwise pooling operations are applied.

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

Maximum and minimum pooling retain extreme activations that may reflect rare object patterns or unusual content, although their specific contributions have not been isolated experimentally.

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

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

Standard deviation pooling summarizes variation among object embeddings.

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

The four pooled blocks are concatenated into a 4,096-dimensional semantic vector.

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

B. Structural Graph Statistics

The structural block adds a compact description of graph size and degree distribution to the semantic vector, but two reference graphs can share these summaries even when their connections differ substantially, which limits what this representation can recover from the original object graph.

Semantic pooling loses explicit connectivity, so eleven coarse graph statistics add node and edge counts together with density, degree summaries and isolated node and leaf node ratios; these summaries do not restore full topology and a reproducible implementation must specify how it handles self loops, graphs with fewer than two nodes, parallel edges and the leaf definition.

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

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

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

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

C. Random Forest Classifier

The Random Forest contains B trees trained on bootstrap samples and considers a random feature subset at each split, where the Gini criterion measures the class impurity of a node sample S .

$$\text{Gini}(S) = 1 - \sum_{c \in \{0,1\}} p_c^2 \quad (15)$$

The predicted label follows majority voting across the trees.

$$\hat{y}_p = \arg\max_{c \in \{0,1\}} \sum_{b=1}^B I(T_b(\phi(G_p)) = c) \quad (16)$$

In this Random Forest implementation, the class score averages tree level class probabilities rather than simply counting hard votes, while any threshold change requires held out validation data because these scores are not necessarily calibrated.

$$\Pr(y_p=1 \mid G_p) = (1 / B) \sum_{b=1}^B \Pr_{\beta}(y_p=1 \mid \phi(G_p)) \quad (17)$$

TABLE I. Proposed semantic graph to vector Random Forest workflow.

Step	Operation	Output
1	Load released object graph with node embeddings and edges	G_p, X_p
2	Separate BERT-65k node vectors from graph topology	X_p, E_p
3	Pool node embeddings: mean, max, min, standard deviation	s_p
4	Compute eleven structural statistics from graph edges	g_p
5	Concatenate semantic and structural feature blocks	$\phi(G_p)$
6	Train Random Forest and evaluate on held out splits	metrics

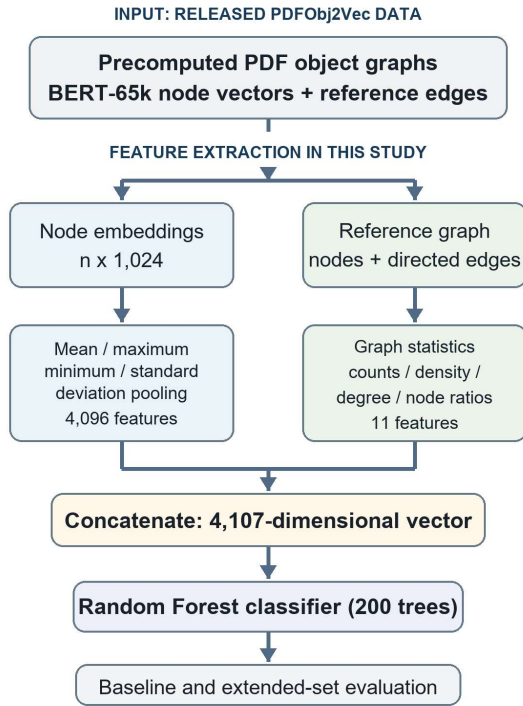


Figure 1 Study pipeline using released PDFObj2Vec graph files and precomputed embeddings.

V. EXPERIMENTAL DESIGN

A. Dataset

The reported training partition contains 6,294 benign graphs and 6,896 malicious graphs, while the baseline test partition contains 2,698 benign graphs and 3,435 malicious graphs and the extended set contains 24,476 benign graphs and 21,471 malicious graphs, so all three partitions contain both classes in substantial numbers.

The experiment uses the released PDFObj2Vec graph dataset with precomputed BERT-65k embeddings [30], and Table II reports the supplied counts, although the available record does not establish whether duplicate files, malware families or source collections overlap across splits and therefore does not justify calling the extended set temporal or independent.

TABLE II. Dataset splits used in this study.

Split	Samples	Benign	Malicious	Purpose
Train	13,190	6,294	6,896	Model fitting
Baseline test	6,133	2,698	3,435	Reported test split
Extended test	45,947	24,476	21,471	Additional evaluation set

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 and TN denote correct malicious and benign classifications while FP and FN denote the corresponding errors, which yield the following evaluation metrics.

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

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

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

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

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

VI. RESULTS AND ANALYSIS

A. Classification Performance

The 200-tree class weighted Random Forest correctly labels all 13,190 training graphs, 6,131 of 6,133 baseline test graphs and 44,154 of 45,947 extended set graphs, although perfect training accuracy does not establish generalisation and the supplied record lacks uncertainty intervals or repeated seed sensitivity.

The baseline test set contains only two missed malicious PDFs and no false positives, whereas the extended set has 1,793 total errors, showing that the very high baseline accuracy should not be treated as the sole summary of generalisation performance.

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
---------------	-------	-------	-------	-------	-------

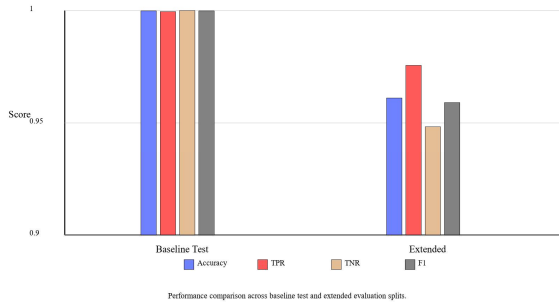


Figure 2 Performance comparison across baseline and extended evaluation splits.

B. Confusion Matrix Interpretation

On the extended set the model produces 23,208 true negatives and 20,946 true positives along with 1,268 false positives and 525 false negatives, so it flags 5.18% of benign files and misses 2.45% of malicious files while achieving 94.29% precision at the dataset prevalence, which may differ in a lower prevalence deployment stream.

The extended false positive rate corresponds to about 52 alerts per 1,000 benign PDFs and the false negative rate corresponds to about 24 missed cases per 1,000 malicious PDFs, although those class specific rates do not predict production alert volume without the prevalence of malicious files in the target environment.

TABLE V. Confusion matrices for all evaluation splits.

Split	TN	FP	FN	TP
Train	6,294	0	0	6,896
Baseline test	2,698	0	2	3,433
Extended test	23,208	1,268	525	20,946

Predicted Benign Predicted Malicious

True Benign

True Malicious

	Predicted Benign	Predicted Malicious
True Benign	23,208 TN	1,268 FP
True Malicious	525 FN	20,946 TP

Extended-set confusion matrix.

Figure 3 Extended set confusion matrix for the Random Forest classifier.

C. Comparison with Graph Neural Classification

Liu and coauthors [29] report 99.93% baseline accuracy and 96.62% extended accuracy for BERT-65k+GIN, whereas this Random Forest reports 99.97% and 96.10%, leaving a numerical extended set gap of 0.52 percentage points that cannot establish equivalence or superiority without matched predictions and paired uncertainty analysis.

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

Model	Baseline accuracy	Extended accuracy	Operational interpretation
BERT-65k + GIN	99.93	96.62	Published aggregate; paired comparison unavailable
BERT-65k pooled vector + Random Forest	99.97	96.10	This study; parsing and embedding costs unmeasured

VII. DISCUSSION

The aggregate results are consistent with pooled embeddings retaining discriminative information, but mean only, four block pooling, structural only and combined feature ablations on identical partitions are needed before assigning credit to any feature group.

The accuracy comparison isolates only the downstream classification result because the Random Forest consumes representations prepared by the PDFObj2Vec pipeline, and any fair cost comparison must include parsing and

embedding time for both approaches before attributing an operational advantage to either classifier.

A fixed length classifier may simplify the final classification stage while the graph parser and BERT-65k embedding stage remain necessary, so unmeasured throughput, memory, failure handling and cost prevent a claim of cheaper or faster end to end deployment.

Grouped permutation importance could compare the mean, maximum, minimum, standard deviation and structural feature blocks, but no such analysis is reported and individual embedding coordinates should not be presented as human readable explanations.

Operational threshold selection requires validation data separate from the final test sets as well as calibrated scores and expected alert volumes at realistic malware prevalence.

VIII. THREATS TO VALIDITY

Internal validity is limited by precomputed embeddings and incomplete records of preprocessing, hyperparameter selection, duplicate handling and split membership, so the feature extraction script, exact graph identifiers and software versions should accompany a reproducible release.

A duplicate or near duplicate file appearing in more than one partition could inflate measured performance even when the classifier is not able to generalise to a genuinely new campaign, so a reproducible split audit should compare file hashes and related families before drawing stronger conclusions from the extended set.

External validity is limited by uncertain family and source overlap as well as document types not represented in the evaluation, while any temporal claim requires timestamps and a chronological split followed by tests on malformed, encrypted, scanned and enterprise generated PDFs.

Adversarial robustness remains untested because no realizable PDF edits or adaptive attacks against this pooled classifier were evaluated, and robustness of the original graph model [29] cannot be transferred without measuring attack success and document validity across the complete raw PDF pipeline.

IX. FUTURE WORK

Audit duplicate hashes and source or malware family overlap across all splits before testing realizable PDF manipulations with validity checks.

Run mean only, four block pooling, structural only and combined feature ablations on identical splits before comparing paired confidence intervals against a rerun GIN baseline.

Measure grouped permutation importance and inspect classification errors without treating individual embedding coordinates as semantic explanations.

Evaluate calibrated thresholds and prevalence sensitive precision using a held out design before testing genuinely timestamped and malware family aware splits.

Measure latency and memory for parsing, embedding, feature extraction and classification both separately and end to end.

Release the code and preprocessing settings together with dataset and split identifiers, prediction scores and software versions where licensing permits.

X. CONCLUSION

This study shows that a Random Forest using 4,107 features formed from pooled PDF object embeddings and graph summaries achieves 99.97% baseline test accuracy and 96.10% extended set accuracy compared with the published BERT-65k+GIN extended accuracy of 96.62% [29], making it a useful classical baseline while split auditing, ablations, calibration and runtime measurement remain necessary before claiming statistical equivalence, adversarial robustness or a deployment advantage.

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, S. Wang, D. She, and S. Jana, "On training robust PDF malware classifiers," USENIX Security Symposium, 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] S. Liu, J. Ming, G. Zhou, X. Liu, J. Fu, and G. Peng, "Analyzing PDFs like Binaries: Adversarially Robust PDF Malware Analysis via Intermediate Representation and Language Model," ACM CCS, 2025, doi: 10.1145/3719027.3744829.
- [30] PDFObj2Vec Evaluation Dataset, Zenodo, version 1, 2025, doi: 10.5281/zenodo.15428814. Accessed: September 2026.