Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.084268

ARTICLE

PE-MILCon: Multiple-Instance Learning with Contrastive Multi-View Representation for Static Windows PE Malware Detection

Tuan Nguyen Kim[1,*], **Son Doan Trung**[1] and **Nguyen Minh Nhut Pham**[2]

[1]Phenikaa School of Computing, Phenikaa University, Duong Noi, Hanoi, Vietnam
[2]Vietnam-Korea University of Information and Communication Technology, The University of Danang, Danang, Vietnam
*Corresponding Author: Tuan Nguyen Kim. Email: tuan.nguyenkim@phenikaa-uni.edu.vn

Received: 22 April 2026; Accepted: 04 June 2026; Published: 13 August 2026

ABSTRACT: Static Windows Portable Executable (PE) malware detection remains a significant challenge due to the growing use of packing, obfuscation, and code reuse techniques, which gradually reduce the effectiveness of signature-based and manually engineered feature approaches. Recent deep learning models that operate directly on binary code or static features have achieved encouraging results; however, most still rely on global file-level representations. Such approaches are susceptible to noise introduced by padding or obfuscation and may overlook localized malicious regions. Moreover, many multi-view methods process different feature sources independently, lacking mechanisms to enforce semantic consistency across views. This paper proposes PE-MILCon, a malware detection framework that integrates Multiple-Instance Learning (MIL) with contrastive multi-view representation learning across raw byte segments and structural PE features. In PE-MILCon, each executable file is modeled as a "bag" of byte segments. The attention mechanism within MIL enables the model to focus selectively on suspicious regions rather than treating the entire file uniformly. In parallel, structural and semantic PE features are encoded as a complementary view. A contrastive loss function aligns the two representations within a shared semantic space, enhancing robustness against obfuscation and packing techniques. The proposed framework operates entirely on static analysis and is trained end-to-end. Experiments on large-scale PE datasets under a strict time-based evaluation protocol show that PE-MILCon achieves an ROC-AUC above 0.98 and an F1-score of approximately 0.96, demonstrating competitive performance compared with existing models. In addition, instance-level attention weights provide intuitive indications of important code regions, supporting malware inspection and forensic analysis. These results suggest that PE-MILCon offers an effective, robust, and interpretable approach for static malware detection.

KEYWORDS: Malware detection; multiple-instance learning; contrastive learning; multi-view learning; static analysis; portable executable; deep learning; cybersecurity

1 Introduction

Recent advances in deep learning have improved the effectiveness of static malware detection. However, challenges related to robustness and generalization in PE malware analysis still remain. This introduction first reviews recent related studies and then presents the motivation and main contributions of PE-MILCon.

1.1 Research Background and Motivation

Malware continues to pose one of the most serious threats to modern computing systems, ranging from enterprise infrastructures and personal devices to cloud platforms. In practice, malicious software

increasingly employs sophisticated techniques such as packing, obfuscation, and polymorphism to evade traditional detection mechanisms. This study focuses on static malware detection for Windows PE files.

Signature-based and rule-driven detection systems, while still useful against known threats, are no longer sufficient to cope with the speed and diversity of contemporary malware evolution. As a result, static analysis approaches based on machine learning and deep learning have become a central component of modern cybersecurity systems [1].

1.2 Characteristics of Modern Malware and Their Impact on Static Detection

In recent years, malware has evolved significantly to evade traditional detection mechanisms. Prominent characteristics of modern malware include: (i) Packing and encryption: much of the binary content is concealed, and malicious logic only becomes visible after unpacking; (ii) Obfuscation and padding: insertion of benign code or dummy data to mislead static analysis and machine learning models; (iii) Code reuse and modularization: many malware families share common components, while malicious behavior is concentrated in specific modules or sections; and (iv) High variability: a single family can generate numerous variants with different byte-level structures but similar behavior.

These characteristics make static malware detection challenging because only a small portion of a PE file may contain decisive malicious information, while the remaining regions act as noise. As a result, global file-level models are often weakened by irrelevant content, motivating approaches such as PE-MILCon that combine local instance-level modeling with multi-view static representations.

1.3 Advances in Machine Learning and Deep Learning for Malware Detection

In recent years, significant progress has been made in applying machine learning and deep learning techniques to malware detection, particularly for Portable Executable (PE) files. The main research directions include: (i) directly analyzing the raw binary byte sequences of executable files using deep neural networks [2]; (ii) exploiting static structural features such as PE headers, import tables, section information, and API calls [3]; and (iii) combining multiple feature sources in a multi-view manner to enhance semantic coverage [4].

These approaches have demonstrated the ability to learn discriminative representations between malicious and benign software, while reducing reliance on manually engineered features. Nevertheless, fundamental limitations remain in how PE data is modeled.

1.4 Limitations of Existing Approaches

Although many current malware detection models achieve strong performance on benchmark datasets, they often rely on simplifying assumptions: (i) Global file-level representation: each executable is mapped to a single vector, even though malicious behavior typically resides in only specific regions or code segments [5]; (ii) Lack of component-level reasoning: models do not differentiate the varying importance of byte segments, sections, or imported functions in the final decision [6]; and (iii) Multi-view without semantic alignment: raw byte features and structural features are usually learned independently and fused only at the final stage, resulting in inconsistent representations [7]. These limitations reduce robustness against packing and obfuscation techniques and constrain the interpretability of model decisions.

1.5 PE-MILCon's Core Idea and Approach

Motivated by the limitations discussed above, this study proposes a different perspective on malware detection. Instead of treating an executable file as a single, indivisible object, we model it as a collection

of smaller components, each contributing differently to the overall malicious behavior. Based on this view, PE-MILCon is built upon two main pillars: (i) Multiple-Instance Learning (MIL) [8], where each PE file is modeled as a "bag" of byte segments, allowing the model to focus on suspicious regions through an attention mechanism; and (ii) Contrastive multi-view representation learning, in which representations derived from raw bytes and structural features are aligned using a contrastive loss function [9], ensuring that both views capture consistent semantic information about file behavior. This combination enables the model to exploit local information while maintaining semantic coherence across different static feature sources.

1.6 Main Contributions of This Paper

This study proposes and implements PE-MILCon, a malware detection framework based on multiple-instance learning and contrastive multi-view representation learning. The primary contributions are as follows: (i) We introduce the PE-MILCon model, in which each PE file is represented as a set of byte segments and processed using an attention-based MIL mechanism to automatically identify suspicious regions; (ii) We design a contrastive multi-view learning scheme that aligns representations derived from raw bytes and PE structural features within a shared semantic space, thereby improving robustness against packing and obfuscation; (iii) We develop an end-to-end static analysis framework that integrates these components into a unified pipeline, trained directly on binary data and structural features without requiring dynamic analysis; and (iv) We conduct comprehensive experiments on large-scale PE datasets, including comparisons with strong machine learning and deep learning baselines, ablation studies [10,11], and time-based evaluation to demonstrate the effectiveness and stability of the proposed model.

Accordingly, the proposed and implemented PE-MILCon framework introduces two main novelties: (i) An architectural design that models each PE file as a set of byte segments and performs instance-level reasoning through attention-based multiple-instance learning (MIL); and (ii) A methodological design that integrates contrastive multi-view learning to align representations derived from raw byte segments and structural PE features within a unified training framework. This combination provides a systematic approach for improving the robustness and interpretability of static malware detection.

2 Related Works

In recent years, malware detection on PE files has shifted from manually engineered features toward end-to-end deep learning models and self-supervised representation learning. Current approaches mainly fall into three categories: (i) Learning directly from raw executable bytes; (ii) Leveraging multiple static features through multi-view modeling or feature fusion; and (iii) Applying contrastive or self-supervised learning to improve generalization. These directions are summarized below.

Deep Learning on Raw PE Bytes: Learning directly from the byte sequence of PE files has become a major direction in static malware detection. MalConv employs a 1D convolutional network to process the entire file at the byte level and reports accuracy in the range of 85%–90%, depending on the dataset. Subsequent studies have shown that global raw-byte models tend to suffer performance degradation under padding or obfuscation [12]. More recent methods, such as MAlign [13], exploit the sequential structure of raw bytes inspired by alignment mechanisms, achieving accuracy above 95% in malware family classification tasks. However, these models predominantly learn global file-level representations and do not perform instance-level reasoning.

Recent studies have also explored advanced deep learning and Transformer-based architectures to further enhance representation learning and robustness in malware detection tasks.

Multi-view and Structural Feature-Based Detection: Many studies exploit static PE features, including headers, import tables, and section metadata. Recent work such as MalSFF [14] combines multiple static feature sources through hybrid feature fusion and multi-architecture learning to improve malware detection performance across heterogeneous PE samples. In traditional machine learning, the EMBER feature set [15] combined with Gradient Boosting attains ROC-AUC ≈ 0.99 on the EMBER benchmark, demonstrating the effectiveness of engineered features. Nevertheless, these approaches typically rely on handcrafted representations or perform fusion outside the core model, without establishing a unified representation learning mechanism across views.

Contrastive and Representation Learning for Malware: Contrastive learning has recently been introduced to enhance generalization. Liu et al. [16] apply SimCLR combined with GRU to PE data, reporting accuracy ≈ 99%, AUC ≈ 98.2%, and F1 ≈ 96.8%. Other self-supervised studies also demonstrate strong performance by learning representations through positive and negative sample pairs. However, these works generally apply contrastive learning within a single view and do not integrate it with multiple-instance learning in static PE settings. In addition, recent work has explored explainable ensemble learning for Android malware detection, demonstrating that combining multiple model perspectives can improve robustness and interpretability in security-oriented AI systems [17].

Multiple-Instance Learning in Malware Analysis: MIL has also been explored in malware research. Stiborek et al. [8] apply MIL to malware classification using behavioral artifacts collected from sandbox execution. However, this approach focuses on dynamic behavioral analysis rather than static executable modeling. In contrast, PE-MILCon formulates static PE malware detection as an MIL problem at the byte-segment level and integrates contrastive multi-view alignment between raw byte segments and structural PE features within a unified framework.

Limitations of Existing Approaches: Overall, current methods exhibit three main limitations: (i) Global raw-byte models lack instance-level reasoning and are sensitive to obfuscation [11,12]; (ii) Multi-view and feature-fusion approaches still mainly rely on global representations or late-stage fusion strategies without explicitly modeling localized malicious regions or semantic consistency across views [14,15]; and (iii) Contrastive learning has not been systematically integrated with multiple-instance learning in the context of static PE analysis [13,16]. This gap motivates a framework that combines local byte-level modeling with multi-view semantic alignment, as realized in PE-MILCon.

In summary, although many methods achieve strong performance, most still rely on global file-level representations without jointly integrating instance-level reasoning and multi-view learning. PE-MILCon is proposed to address this gap.

3 Proposed Model: PE-MILCon

Section 3 presents the proposed PE-MILCon framework, including its overall architecture, multiple-instance learning formulation, contrastive multi-view representation mechanism, and end-to-end processing workflow for static PE malware detection.

3.1 Idea and Solution

Current static malware detection methods typically map an entire executable file into a single global representation, even though malicious behavior often resides only in specific local regions. Such approaches are vulnerable to packing and obfuscation techniques that introduce noise into the file.

The proposed PE-MILCon model represents each PE file as a collection of byte-level instances. Specifically, given an executable file x, it is divided into K consecutive byte segments: $x = \{x_1, x_2, \ldots, x_k\}$,

where each x_i denotes an instance. Malware detection is formulated under MIL framework, where labels are assigned at the file level only.

Each byte segment x_i is encoded using a shared encoder $f_\theta(\cdot)$, producing the representation:

$$h_i = f_\theta(x_i). \tag{1}$$

The file-level representation is then obtained by a weighted aggregation of instance representations z_b, where α_i reflects the contribution of each byte segment:

$$z_b = \sum_{i=1}^{K} \alpha_i h_i. \tag{2}$$

In addition to the raw-byte view, each PE file is also represented by a set of static structural and semantic features, denoted as x^{struct}, which are encoded as:

$$z_s = g_\varnothing(x^{struct}), \tag{3}$$

where $g_\varnothing(\cdot)$ is a learnable encoder that maps structural and semantic features of the PE file, such as header information, section metadata, and import statistics, into a fixed-dimensional representation vector.

PE-MILCon jointly learns the two representations z_b and z_s, while aligning them within a shared semantic space to leverage both local information from raw bytes [18] and global structural information [19]. This design establishes the foundation for contrastive multi-view learning. The full architecture of PE-MILCon is described in the following sections.

3.2 Overall Architecture of PE-MILCon

The PE-MILCon architecture is organized as a pipeline with clearly defined functional blocks, enabling the integration of instance-level reasoning and multi-view representation learning within an end-to-end training framework. Each block serves a specific role, with well-defined inputs and outputs. The components are described as follows:

Block 1: Byte-Instance Encoding

The first block extracts and encodes byte segments from the PE file. The input is the executable file in binary form, which is divided into K consecutive fixed-length byte segments, denoted as $x = \{x_1, x_2, \ldots, x_k\}$. Each segment x_i is treated as an independent instance and processed through a shared encoder $f_\theta(\cdot)$ to obtain an instance-level representation: $h_i = f_\theta(x_i)$; $i = 1, 2, \ldots, K$.

The output of this block is the set of vectors $\{h_i\}$, capturing local information from individual byte segments. This stage is responsible for learning local features from raw bytes, forming the basis for identifying suspicious regions within the PE file.

Block 2: Multi-Instance Aggregation with Attention (MIL Attention Pooling)

The second block aggregates the instance-level representations into a single file-level representation within the multiple-instance learning framework. The input consists of the set $\{h_i\}$ from Block 1. Through a learnable attention mechanism, each instance is assigned a weight α_i, and the file-level representation is computed as: $z_b = \sum_{i=1}^{K} \alpha_i h_i$.

The output of this block is the vector z_b, representing the PE file based on raw bytes. This design enables instance-level reasoning, where byte segments more strongly associated with malicious behavior contribute more significantly to the final decision.

Block 3: Structural PE Feature Encoding

In parallel with the raw-byte branch, this block exploits static structural and semantic features of the PE file.

The input consists of the structural feature set x^{struct}, including information from the PE header, section metadata, and imported functions.

The structural feature representation is constructed from commonly used static PE attributes, including features derived from PE headers, section metadata (e.g., section size and characteristics), and imported functions. These features are encoded and projected into a unified structural feature vector before being processed by the structural encoder. These features are mapped through a dedicated encoder $g_{\varnothing}(\cdot)$ to obtain the structural representation: $z_s = g_{\varnothing}(x^{struct})$.

The output of this block is the vector z_s, which provides a global view of the file's functionality and potential behavior. This branch complements the raw-byte view and is particularly valuable when the binary content is obscured by packing or obfuscation techniques.

Block 4: Contrastive Multi-View Representation Alignment

The fourth block enforces semantic alignment between the two file-level representations produced by the learning branches. The input consists of the pair (z_b, z_s) corresponding to the same PE file, along with representations from other files used as negative samples. Through a contrastive loss function, this block encourages representations from the two views of the same file to be close in feature space, while preserving separability across different files.

During training, negative samples are constructed within each mini-batch. For a positive pair (z_b, z_s) from the same PE file, representations from other files in the same mini-batch are treated as negatives following the standard InfoNCE contrastive learning formulation.

The output of this block is a set of representations adjusted to ensure semantic consistency between raw-byte and structural features.

Block 5: Classification and End-to-End Training

The final block uses the file-level representation z_b (or a fusion of z_b and z_s) to predict the probability of malware vs. benign software. The entire PE-MILCon framework is trained end-to-end on static data, enabling systematic evaluation of each component's role through ablation studies and robustness analysis.

The architecture is modular, with each block serving a clearly defined function and being independently assessable. This design reflects the integration of instance-level reasoning with multi-view representation learning.

To train the PE-MILCon architecture, we employ a joint objective that combines malware classification with multi-view representation alignment. The classification task is formulated as a binary decision problem using binary cross-entropy:

$$L_{cls} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log\left(\widehat{y_i}\right) + (1-y_i)\log\left(1-\widehat{y_i}\right)\right], \tag{4}$$

where y_i ($y \in \{0,1\}$): denotes the ground-truth label; $\widehat{y_i}$ is the predicted probability; and N: the batch size.

To align the raw-byte representation z_b and the structural representation z_s, a contrastive objective is applied:

$$L_{contrastive} = -log\frac{\exp\left(sim(z_b, z_s)/\tau\right)}{\sum_{k=1}^{N}\exp\left(sim(z_b, z_k)/\tau\right)}, \tag{5}$$

where $sim(\cdot,\cdot)$: Cosine similarity; τ: A temperature parameter controlling the sharpness of the distribution; N: The number of samples in the batch (the batch size); z_k: Representations from other samples in the mini-batch used as negative pairs.

The overall training objective of PE-MILCon is defined as

$$L = L_{cls} + \lambda L_{contrastive}, \tag{6}$$

where λ balances the contribution of the contrastive alignment term. The model is optimized jointly in an end-to-end manner.

The contrastive objective follows an InfoNCE-style formulation with a fixed temperature parameter τ; negative samples are drawn from other instances within the same mini-batch, and no additional projection head or stop-gradient mechanism is applied.

To clarify the training objective within the proposed architecture, the contrastive loss is applied during the multi-view representation alignment stage to encourage semantic consistency between the raw-byte representation and the structural-feature representation of the same PE file. During training, this alignment objective is jointly optimized with the classification loss in an end-to-end manner, enabling the model to simultaneously learn discriminative malware representations and maintain consistency across the two complementary views.

To provide a clearer overview of the proposed framework, Fig. 1 illustrates the overall architecture and processing workflow of PE-MILCon.

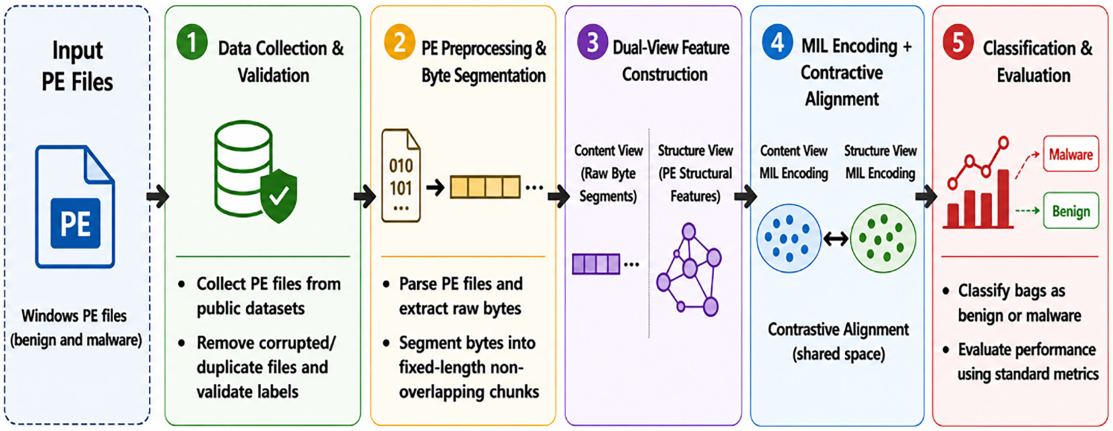

Figure 1: The overall workflow of the proposed PE-MILCon framework.

As shown in Fig. 1, PE-MILCon jointly learns local byte-level representations and structural PE representations, while the MIL attention mechanism enables the framework to focus on suspicious regions that contribute most strongly to the final malware classification decision.

3.3 Byte-Segment Encoder Based on a Lightweight Transformer

In PE-MILCon, each byte segment x_i is treated as a discrete sequence of fixed length $L = 1024$:

$x_i = (b_{i1}, b_{i2}, \ldots, b_{iL}), b_{ij} \in (0, 1, \ldots, 255),$

where b_{ij} denotes the integer value of the byte at position j in segment i, and $i = 1, 2, \ldots, K$, with $K_{\max}$.

Each byte is mapped into a continuous space using a learnable embedding matrix:

$$e_{ij} = Eb_{ij} \in R^{64},\ E \in R^{256\times 64}, \tag{7}$$

where $d_e = 64$ is the embedding dimension.

To preserve positional information, a learnable positional embedding $p_j \in R^{64}$ is added:

$$\tilde{e}_{ij} = e_{ij} + p_j. \tag{8}$$

The segment length is fixed to $L = 1024$ bytes as a practical design choice to balance local context representation and computational efficiency. This size allows each segment to capture meaningful byte patterns and structural cues within a local region of the executable while keeping the sequence length manageable for the lightweight Transformer encoder. Such fixed-length byte chunks are also commonly used in raw-byte malware analysis and sequence-based executable modeling to represent localized regions of PE files [2,13].

Lightweight Transformer Encoder [20]**:** The embedding sequence $(\tilde{e}_{i1}, \tilde{e}_{i2}, \ldots, \tilde{e}_{iL})$ is passed through a lightweight Transformer encoder with the following configuration: Number of layers T: 2; Hidden size d: 128; Attention heads: 4; Feed-forward dimension: 256; Dropout: 0.1.

In each layer, multi-head self-attention is computed as:

$$Attention\,(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{9}$$

Here: $Q = XW_Q, K = XW_K, V = XW_V$; $W_Q, W_K, W_V \in R^{d\times d}$ are learnable parameter matrices; $d_k = \frac{d}{H} = 32$ (do 4 heads × 32 = 128); X denotes the input to the corresponding layer.

Each layer consists of: Multi-head attention; Residual connection; Layer normalization; Feed-forward network (128 to 256 down to 128); Dropout (0.1).

After two layers, the contextualized representations are obtained:

$H_i = (h_{i1}, h_{i2}, \ldots, h_{iL}),\ h_{ij} \in R^{128}$.

Segment-Level Aggregation: The representation of a byte segment is computed using mean pooling:

$$h_i = \frac{1}{L}\sum_{j=1}^{L} h_{ij}, h_i \in R^{128}. \tag{10}$$

The vector h_i represents the i-th byte segment and is fed into the MIL attention mechanism (Sections 3.2 and 3.3).

This configuration is selected based on the following criteria: Sufficient capacity to model dependencies within a 1024-byte segment; Moderate parameter size to enable training with large $K_{\max}$; Reduced risk of overfitting under time-based evaluation; Compatibility with the hardware resources described in Section 4.1.

With a hidden size of 128 and two layers, the byte-segment encoder maintains a moderate number of parameters, allowing end-to-end training without significantly increasing computational cost.

3.4 Processing Pipeline of PE-MILCon

The processing workflow of PE-MILCon is organized as a static pipeline, where each PE file is passed through two parallel branches before being merged for final classification:

Step 1 (PE File Segmentation): The executable file in binary form is directly read and divided into consecutive byte segments of fixed length $L = 1024$. For each file, up to $K_{\text{Max}} = 1024$ segments are used; if fewer segments are available, zero-padding is applied; if more, segments are truncated in order of appearance. In this study, the maximum number of segments is limited to $K_{\text{Max}} = 1024$ to maintain a manageable computational cost within the multiple-instance learning framework. Since each segment has a fixed length of 1024 bytes, this configuration allows the model to process approximately the first 1 MB of each executable file. In practice, this portion typically contains the most informative regions of PE files, including headers, code sections, and commonly used imports, while keeping the training process computationally efficient.

Step 2 (Byte Segment Encoding): Each segment x_i is mapped through a Lightweight Transformer encoder $f_\theta(\cdot)$ to obtain an instance-level representation: $h_i = f_\theta(x_i) \in R^{128}$.

Step 3 (File-Level Aggregation via MIL): The set of representations $\{h_i\}$ is aggregated into a file-level representation z_b using a learnable attention mechanism: $z_b = \sum_{i=1}^{K} \alpha_i h_i$.

Step 4 (Structural Feature Extraction): In parallel with the raw-byte branch, static features from the PE header, section metadata, and import table are extracted and encoded as: $z_s = g_\varnothing(x^{struct})$.

Step 5 (Multi-View Representation Alignment): The two representations z_s and z_b are aligned within a shared feature space using a contrastive learning mechanism to enhance semantic consistency between the two views.

Step 6 (Classification): The final file-level representation is passed through a fully connected layer to estimate the probability:

$$\hat{y} = \sigma\left(Wz + b\right), \tag{11}$$

where z denotes the fused representation; σ is the sigmoid function; W and b are the parameters of the classification layer, optimized jointly with the entire model during end-to-end training.

During training, the model is optimized end-to-end using a joint objective that combines the classification loss and the contrastive alignment loss (Eq. (6)):

$$\mathrm{L} = L_{cls} + \lambda L_{contrastive},$$

where L_{cls} denotes the classification loss, L_{con} is the contrastive loss encouraging alignment between z_b and z_s, and λ controls the balance between the two terms.

The PE-MILCon workflow consists of two parallel branches (raw bytes and structural features) that are merged at the file-representation level, enabling simultaneous exploitation of local and global information. The entire pipeline is trained end-to-end on static data, facilitating systematic evaluation of each component through ablation and robustness analyses.

Section 3 presents the core ideas and architecture of PE-MILCon, including multiple-instance learning formulation, the Lightweight Transformer encoder, and the overall processing pipeline. Components such as attention weights α_i, the structural encoder $g_\varnothing(\cdot)$, contrastive multi-view learning, and the objective function are integrated into the end-to-end training process and further clarified in the experimental section.

4 Experiments and Results

This section presents the experimental setup, evaluation protocol, and performance analysis of PE-MILCon under time-based malware detection scenarios. The experiments are designed to assess detection accuracy, robustness, and interpretability under realistic PE malware conditions.

4.1 Experimental Setup

- **Dataset Construction and Time-Based Split:** The dataset consists exclusively of Windows Portable Executable (PE) files, including both malware and benign software samples. Malware samples were collected from MalwareBazaar [21], and only valid PE32/PE32+ executables that could be successfully parsed were retained. Samples with corrupted headers, incomplete metadata, or duplicated SHA-256 hashes were removed during preprocessing. The temporal information of malware samples was determined using the first_seen field provided by MalwareBazaar and was used for the time-based train/validation/test split.

Benign samples were collected from officially distributed Windows applications and common third-party software packages with valid digital signatures. Similar preprocessing and PE validation procedures were applied to the benign set. The temporal information of benign files was obtained from signature metadata or recorded collection timestamps to maintain consistency with the time-based evaluation protocol.

To reflect real-world deployment and prevent temporal leakage, the data are split using a time-based protocol. In this protocol, the model is trained on samples collected in earlier periods and evaluated on samples collected later. Although certain malware families may appear across multiple time periods, the temporal separation of samples reduces potential information leakage and provides a more practical assessment of the model's ability to detect emerging malware: (i) Samples from 01/2021 to 12/2023 are used for training and validation, totaling 48,000 samples; (ii) Samples from 01/2024 to 06/2024 are reserved exclusively for testing, comprising 6000 samples. Let t_1 = 31/12/2023; then the train/validation set includes samples with $t \leq t_1$, while the test set includes samples with $t > t_1$.

The dataset follows the time-based split described above, with 38,400 samples used for training, 9600 samples for validation, and 6000 samples reserved for testing. This configuration ensures that model training and evaluation are performed on temporally separated data.

There is no SHA-256 overlap between the two sets; duplicate or highly similar files are removed prior to splitting. Within the 48,000 training/validation samples, data are further divided into 80% for training and 20% for validation. Labels are assigned at the file level with two classes: malware and benign.

This setup ensures that the model is trained on historical data and evaluated on future samples, aligning with real-world deployment scenarios and enabling analysis of temporal performance degradation.

To illustrate the diversity of the collected malware samples, Table 1 lists the most frequent malware families observed in the dataset.

Table 1: Top malware families in the dataset.

	Red-Line	Agent-Tesla	Form-Book	Smoke-Loader	Lokibot	Vidar	SnakeKey-Logger	Remcos	Nano-Core	Async-RAT
Samples	3412	2987	2514	2102	1874	1663	1542	1318	1207	1096

These families represent a substantial portion of the collected malware samples and reflect the diversity of threats included in the dataset.

- **Raw Byte Preprocessing and Instance Construction:** For the raw-byte branch, each PE file is read up to a maximum of 1 MB from the beginning of the file to ensure consistency across samples and to control computational cost. This limit is uniformly applied to all deep learning models, including MalConv and multi-view variants, to guarantee fair comparison. The resulting byte stream is then divided into consecutive fixed-length segments of L = 1024 bytes. Each segment corresponds to an instance in the

multiple-instance learning framework. The maximum number of instances per file is capped at $K_{\max} = 1024$. If the number of segments exceeds this limit, excess segments are truncated; if fewer segments are available, the file is padded with zeros at the end. After preprocessing, each file is represented as a set $x = \{x_1, x_2, \ldots, x_k\}$; $K \leq K_{\max}$, which serves as the direct input to the byte encoder and the MIL attention mechanism (described in Sections 3.2 and 3.3).

This setup enables the model to learn local representations at the byte-segment level while mitigating the influence of irrelevant regions within the PE file, particularly in scenarios involving padding or packing.

- **Extraction and Encoding of PE Structural Features:** In addition to the raw-byte branch, we extract static structural features from each PE file using a PE analysis library (e.g., pefile), relying solely on static analysis. The feature set x^{struct} consists of three main groups: (i) PE header features [15], such as image size, number of sections, subsystem, and entry point; (ii) Section statistics [15], including size, entropy, and execution permissions, aggregated at the file level; and (iii) Import-based statistics [15], such as the number of DLLs, number of imported APIs, and related summary statistics. One-hot encoding of APIs is avoided to prevent high dimensionality and potential data leakage.

All features are normalized using z-score statistics computed on the training set and then fixed for the test set. The resulting feature vector is passed through a structural encoder $g_{\varnothing}(\cdot)$ (a multi-layer MLP) to produce the representation z_s, aligned in dimensionality with the raw-byte branch to support contrastive multi-view learning (Sections 3.2 and 3.3).

This design enables semantic modeling at the file level without executing the code, while enhancing robustness against packing and obfuscation.

- **Comparison Methods and Ablation Setup:** To objectively evaluate PE-MILCon, we select baselines representing major directions in static malware detection, each corresponding to specific components of the proposed architecture: (i) LightGBM-EMBER [15]: A traditional machine learning model trained on EMBER-style handcrafted static features. This baseline represents the feature engineering paradigm and does not utilize raw bytes, enabling comparison between handcrafted features and end-to-end representation learning; (ii) MalConv [2]: A deep learning model operating on raw bytes, processing the entire file as a single sequence to produce a global file-level representation, without multiple-instance learning. Comparing with MalConv allows us to assess the benefit of the MIL mechanism in PE-MILCon; (iii) Dual-View Concat [14]: A multi-view model that combines raw bytes and structural features by concatenating their representations before classification. The architecture includes a MalConv branch and an MLP [22], but does not employ MIL attention or contrastive learning. This baseline enables isolated evaluation of the semantic alignment mechanism introduced in PE-MILCon.

For the LightGBM-EMBER baseline, the feature representation follows the standard EMBER feature specification introduced in the original EMBER dataset (2018), ensuring consistency with the commonly used benchmark configuration.

Ablation Variants of PE-MILCon: To analyze the role of each component, we construct the following variants: (i) Without MIL: Replacing attention pooling with global average pooling; (ii) Without Contrastive: Removing the contrastive multi-view learning component; (iii) Raw-only: Using only the raw-byte branch; (iv) Struct-only: Using only the structural branch.

These variants allow us to disentangle the effects of (a) instance-level reasoning and (b) contrastive multi-view representation learning, the two core mechanisms underlying PE-MILCon.

- **Training Protocol:** All models (PE-MILCon and baselines) are trained under a unified protocol to ensure fairness: (i) The same time-based dataset split; (ii) Identical preprocessing for raw bytes and

structural features; (iii) The same train/validation split and early stopping strategy; (iv) The same number of seeds and reporting format (mean ± standard deviation); and (v) The same evaluation metrics on an independent test set.

- PE-MILCon: All components (byte encoder, MIL attention, structural encoder, and contrastive module) are jointly optimized end-to-end using the objective function described in Sections 3.2 and 3.3.
- Optimization: Adam optimizer [23] with an initial learning rate of 1×10^{-3}, batch size of 64, and a maximum of 50 epochs. A ReduceLROnPlateau scheduler reduces the learning rate by a factor of 0.5 if validation F1 does not improve for three consecutive epochs.
- Early stopping: Based on validation F1 with patience = 5; the best-performing model is saved for test evaluation. Hyperparameters (including the weighting factor λ) are selected on the validation set and kept fixed thereafter.
- Seeds and reproducibility: Each experiment is repeated with five different seeds; seeds are fixed for NumPy [24] and PyTorch [25]; deterministic settings are enabled when possible. Final results are reported as mean ± standard deviation on the time-based test set.
- Baseline fairness: MalConv and Dual-View Concat use the same batch size, number of epochs, scheduler, and early stopping strategy as PE-MILCon. LightGBM-EMBER is tuned on the validation set and evaluated with fixed hyperparameters on the test set.

- **Evaluation Metrics and Testing Scenarios:** Performance is assessed using Accuracy, Precision, Recall, F1-score [26], and ROC-AUC [27], computed on the time-based test set and reported as mean ± standard deviation across five seeds. Precision and Recall reflect false alarm control and detection capability, respectively; F1-score is used for model selection; ROC-AUC measures overall discriminative ability. Beyond overall evaluation, we conduct: (i) Time-based generalization analysis [28] to measure performance degradation over time; (ii) Robustness analysis [29] on datasets augmented with padding or high-entropy samples; and (iii) Interpretability analysis through the distribution of attention weights α_i and identification of high-contribution byte segments.

All evaluations are performed using a unified, standardized script to ensure consistency and provide a comprehensive assessment of accuracy, stability, and real-world applicability of PE-MILCon.

- **Experimental Environment:** Experiments are conducted in an isolated environment on a Linux server (Ubuntu 22.04) equipped with an NVIDIA GPU (24 GB VRAM), Intel Xeon CPU, and 128 GB RAM. The models are implemented in Python 3.10 using PyTorch; LightGBM-EMBER is implemented with the official LightGBM library.

To ensure reproducibility, all preprocessing, training, and evaluation steps are executed through unified scripts; random seeds are fixed; normalization statistics are computed solely on the training set and consistently applied to the test set. Malware samples are not publicly distributed; instead, hash lists and data collection scripts are provided to enable dataset reconstruction. The pipeline is designed for controlled reproducibility while adhering to safe malware handling practices.

4.2 Experimental Results

Before presenting the experimental results, we briefly discuss the computational characteristics of the proposed architecture. PE-MILCon employs a lightweight Transformer encoder with only two layers and a hidden size of 128, resulting in a moderate parameter size suitable for practical deployment. Compared with models that process the entire binary as a single sequence (e.g., MalConv), the segment-based design enables efficient instance-level processing while maintaining scalability for large PE files.

4.2.1 Overall Comparison with Baseline Methods

This section presents a comparison between PE-MILCon and the baselines (Section 4.1) under a unified experimental setting: 48,000 training/validation samples and 6000 time-split test samples, using the same training protocol and evaluation criteria. Results are reported as mean ± standard deviation across multiple seeds (5 seeds).

Due to differences in datasets and evaluation protocols, direct comparison with previously published results may be unreliable, particularly since many prior studies adopt IID splits rather than time-based evaluation. Therefore, we re-implemented representative baselines and evaluated all models under the same time-based split and unified training protocol (Section 4) to ensure fairness and reproducibility.

Result analysis (**from Table 2**): (i) LightGBM-EMBER achieves a high ROC-AUC (97.6%) but a relatively lower F1-score (91.8%), indicating the limitations of handcrafted features under time-based evaluation; (ii) MalConv improves the F1-score to 93.7% by directly leveraging raw bytes, yet its global representation makes the model sensitive to noise and padding; (iii) Dual-View Concat reaches an F1-score of 94.7%, demonstrating the benefit of combining raw bytes and structural features, but it lacks instance-level reasoning and semantic alignment between the two views; (iv) PE-MILCon achieves an F1-score of 96.1% and a ROC-AUC of 99.0%, outperforming Dual-View Concat by 1.4% in F1 while also exhibiting the lowest standard deviation (±0.4).

Table 2: Performance comparison on the time-based test set.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
LightGBM-EMBER	92.8 ± 0.4	93.5 ± 0.6	90.2 ± 0.8	91.8 ± 0.5	97.6 ± 0.3
MalConv	94.1 ± 0.5	94.8 ± 0.7	92.6 ± 0.9	93.7 ± 0.6	98.1 ± 0.4
Dual-View Concat	95.0 ± 0.4	95.6 ± 0.5	93.8 ± 0.7	94.7 ± 0.5	98.5 ± 0.3
PE-MILCon	96.3 ± 0.3	96.9 ± 0.4	95.4 ± 0.6	96.1 ± 0.4	99.0 ± 0.2

This improvement aligns with the model design: The MIL attention mechanism mitigates the influence of irrelevant byte segments, while contrastive multi-view learning enhances semantic consistency between raw-byte and PE structural representations under time-based evaluation. The corresponding comparative and ablation results are summarized in Tables 2 and 3, respectively.

Table 3: Ablation analysis of PE-MILCon.

Model	Accuracy (%)	F1-Score (%)	ROC-AUC (%)
Struct-only	91.5 ± 0.6	90.3 ± 0.7	97.2 ± 0.4
Raw-only (Global pooling)	94.0 ± 0.5	93.6 ± 0.6	98.1 ± 0.4
PE-MILCon w/o Contrastive	95.4 ± 0.4	94.8 ± 0.5	98.6 ± 0.3
PE-MILCon w/o MIL	95.2 ± 0.5	94.6 ± 0.6	98.5 ± 0.3
PE-MILCon (full)	96.3 ± 0.3	96.1 ± 0.4	99.0 ± 0.2

4.2.2 Ablation Analysis

To quantify the contribution of each component in PE-MILCon, we constructed the ablation variants described in Section 4.1 and trained them under the same time-based split, optimization protocol, and number of seeds. Results are reported as mean ± standard deviation on the test set.

Although time-based evaluation introduces additional variability due to temporal malware drift, PE-MILCon consistently achieves the lowest standard deviation across all evaluated metrics.

- **Contribution analysis of each component (from Table 3):**

(i) Contribution of the structural branch: The Struct-only model achieves an F1-score of 90.3%, significantly lower than models that exploit raw bytes. This indicates that PE structural features contain useful information but are insufficient on their own to achieve high detection performance.

(ii) Contribution of multiple-instance learning (MIL): Comparing: Raw-only (global pooling): F1 = 93.6%; PE-MILCon w/o MIL: F1 = 94.6%; PE-MILCon (full): F1 = 96.1%.

Replacing global pooling with MIL attention improves the F1-score by approximately +1.5% compared to the variant without MIL. This suggests that instance-level reasoning enables the model to focus on informative byte segments while reducing the impact of padding and irrelevant regions.

(iii) Contribution of contrastive multi-view learning: Comparing: PE-MILCon w/o Contrastive: F1 = 94.8%; PE-MILCon (full): F1 = 96.1%.

The contrastive component yields an additional improvement of about +1.3% in F1-score and increases ROC-AUC by 0.4%. This demonstrates that enforcing semantic alignment between raw-byte and structural representations helps the model learn more stable representations under the time-based evaluation setting.

Combined effect: Notably, removing either MIL or the contrastive component leads to a performance drop compared to the full model. This indicates that the two mechanisms not only contribute independently but also reinforce each other in improving generalization.

The ablation analysis confirms that: (i) MIL attention plays a substantial role in focusing on informative byte regions; (ii) Contrastive multi-view learning enhances semantic consistency between the two representations; and (iii) Combining both mechanisms results in an overall F1-score improvement of approximately +1.4% to +2.5% compared to variants lacking one of the components.

These findings support the necessity of the PE-MILCon design and clarify the source of the performance gains observed in Table 2.

- **Ablation Analysis Plot of PE-MILCon:**

To visualize the contribution of each component in PE-MILCon, Fig. 2 illustrates the changes in Accuracy, F1-score, and ROC-AUC across different model variants. The plot shows the progression from simpler configurations (single-branch models) to the full architecture.

As shown in Fig. 2, performance improves steadily as the MIL mechanism and contrastive multi-view learning are incorporated. The full PE-MILCon model achieves the highest values across all three metrics, indicating that these two components play complementary roles in enhancing discriminative capability and generalization performance.

4.2.3 Robustness Analysis under Padding and High-Entropy Conditions

To evaluate robustness under realistic perturbation conditions, we selected two representative scenarios commonly associated with evasion behaviors in Windows malware. The first scenario, artificial padding, simulates the insertion of irrelevant bytes into PE files to dilute malicious patterns and disrupt raw-byte feature extraction. The second scenario, high-entropy perturbation, approximates packing and obfuscation techniques frequently used to conceal executable content and alter statistical properties of malware binaries. These two perturbation types are widely recognized as practical challenges for static malware analysis and raw-byte-based detection models [5,7,12].

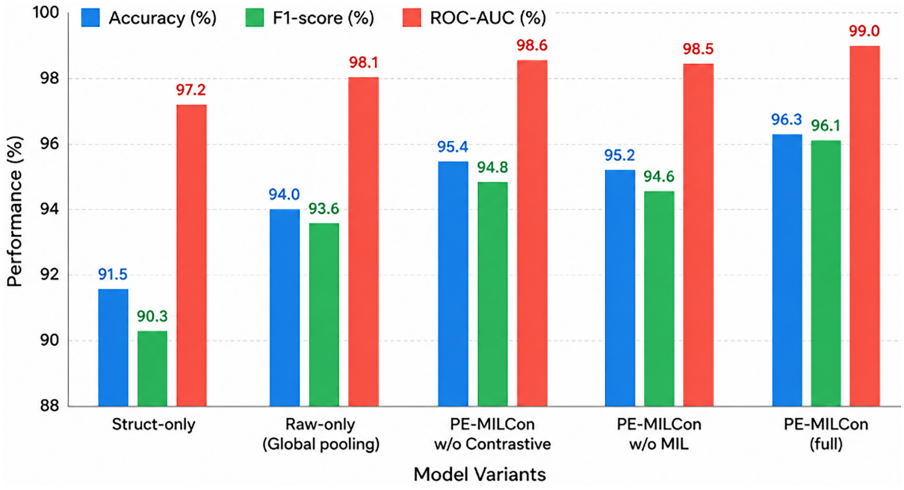

Figure 2: Ablation contribution analysis across accuracy, F1-score, and ROC-AUC.

To evaluate model stability under common real-world perturbations, we consider two additional testing scenarios: (i) Appending padding to the end of PE files; and (ii) Evaluating on a subset of samples with high entropy. For the padding experiment, zero bytes are appended to the end of each executable file to simulate common padding patterns observed in packed binaries, while structural features remain unchanged. High-entropy samples are defined as binaries with entropy greater than 7.0, which typically correspond to packed or obfuscated files. The results are presented in Table 4a,b.

- **Impact of padding:** Table 4a shows that when 50 KB of padding is added, MalConv's F1-score decreases from 93.7% to 89.8% (ΔF1 = −3.9%), while Dual-View Concat drops by 2.3%. In contrast, PE-MILCon declines by only 1.1% (from 96.1% to 95.0%). Under larger padding (+200 KB), the differences become more pronounced: MalConv drops by 8.2% F1 (93.7% down to 85.5%); Dual-View Concat decreases by 5.4%; LightGBM-EMBER declines by 4.4%; whereas PE-MILCon reduces by only 2.5% (96.1% down to 93.6%).

Table 4: (a) F1-score robustness under perturbation scenarios; (b) ROC-AUC robustness under perturbation scenarios.

(a)

Model	F1-Original	F1-+50 KB	ΔF1	F1-+200 KB	ΔF1	F1-High Entropy	ΔF1
LightGBM-EMBER	91.8	89.6	−2.2	87.4	−4.4	88.9	−2.9
MalConv	93.7	89.8	−3.9	85.5	−8.2	90.1	−3.6
Dual-View Concat	94.7	92.4	−2.3	89.3	−5.4	92.0	−2.7
PE-MILCon	96.1	95.0	−1.1	93.6	−2.5	94.8	−1.3

(b)

Model	AUC-Original	AUC-+50 KB	ΔAUC	AUC-+200 KB	ΔAUC	AUC-High Entropy	ΔAUC
LightGBM-EMBER	97.6	96.8	−0.8	95.9	−1.7	96.5	−1.1
MalConv	98.1	96.0	−2.1	93.4	−4.7	96.8	−1.3
Dual-View Concat	98.5	97.3	−1.2	95.6	−2.9	97.8	−0.7
PE-MILCon	99.0	98.6	−0.4	97.9	−1.1	98.4	−0.6

A similar trend is observed for ROC-AUC (Table 4b). At +200 KB padding, MalConv loses 4.7 AUC points; Dual-View Concat drops by 2.9 points; and PE-MILCon decreases by only 1.1 points.

The relatively small degradation of PE-MILCon suggests that the MIL attention mechanism mitigates the impact of injected byte segments by assigning lower weights to instances containing padding, rather than treating the entire file uniformly as in global pooling models.

- **Performance under high-entropy conditions:** On the high-entropy test set, performance differences among models remain evident: MalConv shows a 3.6% drop in F1; LightGBM-EMBER decreases by 2.9%; Dual-View Concat drops by 2.7%; whereas PE-MILCon declines by only 1.3% (from 96.1% to 94.8%). In terms of ROC-AUC, PE-MILCon decreases by 0.6 points, smaller than the reductions observed for MalConv (−1.3) and LightGBM-EMBER (−1.1). These results suggest that combining the structural branch with contrastive multi-view learning helps stabilize representations when local byte distributions are significantly altered due to packing or encryption.
- **Overall degradation comparison:** Across all perturbation scenarios, the F1 degradation (ΔF1) of PE-MILCon remains below 3% under every test condition. In contrast, MalConv experiences up to an 8.2% drop under large-padding scenarios, and Dual-View Concat also shows notable degradation as padding increases, although it remains more stable than MalConv. These findings indicate that the two core components of PE-MILCon play complementary roles: MIL mitigates the impact of local noise, while contrastive multi-view learning enhances the stability of file-level representations.
- **Performance degradation plots:** To visualize performance drops under different perturbation scenarios, Fig. 3a presents the F1-scores of the models on the original test set, the padded sets (+50 and +200 KB), and the high-entropy set. The plot enables direct comparison of the relative degradation across methods.

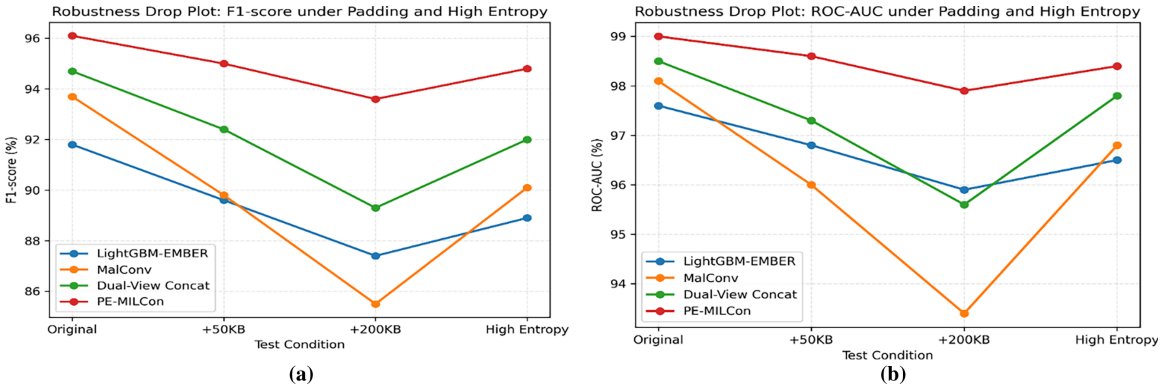

Figure 3: (**a**) Robustness drop plot (F1-score) under padding and high-entropy conditions; (**b**) Robustness drop plot (AUC) under padding and high-entropy conditions.

Fig. 3a shows that PE-MILCon exhibits smaller performance drops under both padding and high-entropy conditions. This observation is consistent with the quantitative results in Table 4a, indicating that the MIL mechanism and contrastive multi-view learning contribute to improved stability against common structural perturbations.

In addition to F1-score, we further analyze degradation in terms of ROC-AUC to assess overall class separability under perturbations. Fig. 3b illustrates the ROC-AUC changes of the models on the original and modified test sets (Table 4b).

As shown in Fig. 3b, PE-MILCon maintains smaller ROC-AUC reductions than the baselines in both padding and high-entropy scenarios. These findings align with the F1-score analysis and reinforce the

conclusion that combining MIL with contrastive multi-view learning enhances the robustness of file-level representations.

- **Conclusion from robustness analysis:** Based on Table 4a,b, PE-MILCon not only achieves strong performance on the original test set but also demonstrates smaller degradation than the baselines under common perturbations. This suggests greater practical deployment potential in real-world environments where malware is often packed or obfuscated to evade detection systems. The corresponding degradation trends are further illustrated in Fig. 3a,b.

4.2.4 Attention-Based Interpretability Analysis

An important advantage of the multiple-instance learning framework in PE-MILCon is its ability to analyze the relative contribution of each byte segment through the attention weights α_i. Unlike models that rely on global pooling, the attention mechanism enables explicit quantification of the role of each instance x_i in the file-level classification decision.

- **Attention weight distribution:** For each sample in the time-split test set, we collect the distribution of attention weights α_i across all instances. Empirical observations indicate that attention is typically concentrated on a small subset of byte segments, while most instances receive weights close to zero. This pattern suggests that the model does not treat the entire file uniformly, but instead prioritizes regions containing more discriminative information. To quantify this concentration, we compute the proportion of total attention mass assigned to the $top\text{-}k$ instances (e.g., $k = 5$ or $k = 10$). This metric captures the degree of attention focus and allows comparison between malware and benign samples.
- **Comparison of α_i distribution between Malware and Benign:** To quantify the concentration level of the attention mechanism in PE-MILCon, we compute the cumulative weight assigned to the $top\text{-}k$ instances with the highest α_i values for each sample. Table 5 reports Mass@k on the time-split test set, presented as mean ± standard deviation across five independent training seeds.

Table 5: Top-k attention mass (mean ± std over 5 seeds).

k	Malware (mean ± std)	Benign (mean ± std)
1	0.30 ± 0.02	0.14 ± 0.01
3	0.54 ± 0.03	0.27 ± 0.02
5	0.69 ± 0.03	0.40 ± 0.03
10	0.83 ± 0.02	0.57 ± 0.03
20	0.92 ± 0.01	0.75 ± 0.02

The results indicate that malware samples tend to exhibit significantly stronger attention concentration than benign ones. Specifically, at $k = 5$, malware achieves Mass@5 ≈ 0.69, whereas benign samples reach only about 0.40. The small standard deviation across different seeds suggests that this attention pattern is stable and not strongly influenced by random initialization.

To visualize the accumulation of attention weights as the number of instances increases, Fig. 4 presents the Mass@k curves corresponding to the values reported in Table 5. Error bars represent the standard deviation over five seeds, reflecting the consistency of the attention mechanism during training.

As shown in Fig. 4, for malware samples, the cumulative attention weight rises rapidly with increasing k, with the top-5 instances accounting for approximately 60%–70% of the total weight. In contrast, benign samples exhibit a more dispersed attention distribution and require a larger number of instances to reach a comparable cumulative mass. This suggests that classification decisions for malware are typically driven by

a small number of highly discriminative byte segments, whereas benign files do not concentrate strongly on specific local regions. This observation further supports the role of the MIL mechanism in identifying and exploiting informative regions within PE files.

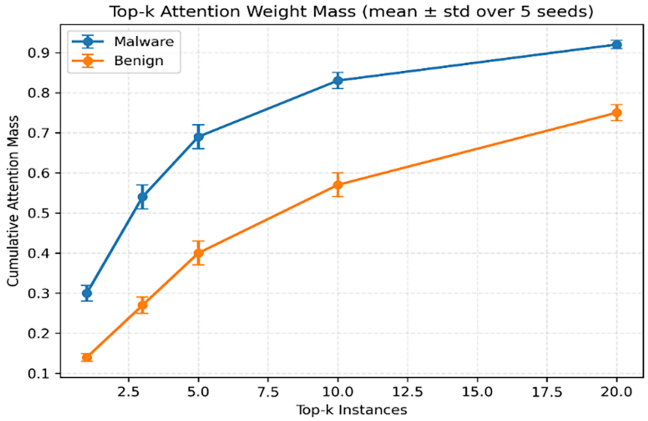

Figure 4: Top-k attention weight mass (mean ± std over 5 seeds).

The tendency of malware samples to concentrate attention on a limited set of instances also helps explain the robustness results reported in Section 4.2.3. Because noisy segments introduced by padding or high-entropy transformations are assigned lower weights during inference, PE-MILCon maintains smaller performance degradation under such perturbations.

- **Linking attention to PE structure:** To enhance interpretability, each byte segment x_i is mapped back to its corresponding offset within the PE file, allowing identification of the associated section (e.g., .text, .data, or overlay). The analysis shows that high-weight instances frequently fall into: executable sections; high-entropy regions; or areas related to imports and the entry point. This observation aligns with security intuition, as executable or packed regions often contain malicious behavior.
- **Comparison with the non-MIL variant:** Compared with the global average pooling variant, MIL attention not only improves performance (Section 4.2.2) but also enables local contribution analysis, supporting malware inspection and forensic analysis.
- **Practical implications:** Although attention analysis does not provide definitive causal evidence, it highlights the relative importance of byte regions in model decisions. Identifying high-weight byte segments can support manual malware analysis, detect anomalous PE regions, and improve model transparency, further reinforcing the role of MIL in PE-MILCon.

5 Discussion

This section summarizes the experimental findings and clarifies the role of each component in PE-MILCon for improving temporal generalization, robustness, and interpretability in static malware detection.

Comparison with Recent Malware Detection Approaches: Recent malware detection studies have explored raw-byte modeling [2], multi-static feature fusion and hybrid malware representation learning [14], adversarial robustness analysis [12], and contrastive representation learning [16]. However, most existing methods still rely mainly on global file-level representations or late-stage feature fusion.

In contrast, PE-MILCon performs instance-level reasoning through multiple-instance learning and aligns raw-byte and structural PE representations within a shared semantic space using contrastive multi-view learning. In addition, robustness analysis under artificial padding and high-entropy perturbations shows that PE-MILCon exhibits smaller performance degradation than the baseline methods.

Overall performance and time-based generalization: Results in Table 2 indicate that PE-MILCon achieves an F1-score of 96.1% and a ROC-AUC of 99.0% on the time-split test set, outperforming the Dual-View Concat model (94.7%) by 1.4% F1 and MalConv (93.7%) by 2.4%. Compared with the traditional ML baseline LightGBM-EMBER (F1 = 91.8%), the improvement reaches approximately 4.3%. Importantly, these results are obtained under a strict time-based split, reflecting the model's ability to adapt to samples appearing after the training period. The performance gap between validation and test remains limited, suggesting that the model does not over-rely on the training data distribution.

Role of multiple-instance learning: The ablation analysis (Table 3) shows that removing the MIL attention mechanism reduces the F1-score from 96.1% to 94.6%, corresponding to a decrease of about 1.5%. This confirms that instance-level reasoning enables the model to focus on relevant byte segments while mitigating the influence of irrelevant regions. Robustness results in Table 4a further indicate that the F1 degradation of PE-MILCon under the +200 KB padding scenario is significantly smaller than that of MalConv, supporting the hypothesis that attention-based MIL contributes to improved resilience against padding transformations.

Role of contrastive multi-view learning: When the contrastive component is removed, the F1-score decreases from 96.1% to 94.8%, corresponding to a drop of approximately 1.3%. This suggests that aligning raw byte representations with structural PE features enhances semantic consistency and strengthens time-based generalization. The simultaneous improvement in both F1-score and ROC-AUC indicates that contrastive learning not only balances precision and recall but also enhances overall class separability.

Robustness and transparency: Robustness analysis shows that PE-MILCon exhibits smaller ΔF1 degradation than the baselines under padding and high-entropy scenarios, indicating that the model learns more stable representations rather than relying on superficial raw-byte signals. Attention analysis identifies byte segments with high weights, often associated with executable sections or high-entropy regions, providing useful cues for malware analysis.

Thus, combining MIL and contrastive multi-view learning improves detection performance, while the time-based evaluation and attention mechanism demonstrate stronger robustness and interpretability.

6 Conclusion and Future Work

This paper proposed the PE-MILCon framework for static malware detection on PE files. The proposed architecture integrates three key components: (i) Byte-segment encoding under a multiple-instance learning paradigm, (ii) Structural feature encoding of PE files, and (iii) Contrastive multi-view representation learning to enhance semantic consistency between the two information sources. Experimental results under a strict time-based split demonstrate that PE-MILCon achieves an F1-score of 96.1% and a ROC-AUC of 99.0%, outperforming the re-implemented baselines under the same protocol. Ablation analysis confirms that both the MIL mechanism and the contrastive learning component contribute significantly to the overall performance. Furthermore, robustness evaluations and attention analysis indicate that the model maintains stable performance under padding and high-entropy transformations, while providing instance-level interpretability at the byte-segment level.

Future work includes integrating dynamic behavioral features, improving generalization under out-of-distribution conditions, and optimizing the architecture for lower computational cost and large-scale deployment.

Acknowledgement: The authors would like to thank Phenikaa University for its support.

Funding Statement: This research was funded by Phenikaa University.

Author Contributions: Tuan Nguyen Kim: Proposed and developed the proposed model; Nguyen Minh Nhut Pham. Ensured the mathematical basis for the proposed model; Son Doan Trung: Experimental and results analysis. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Malware samples were collected from MalwareBazaar (accessed on 14 May 2026). The data collection and preprocessing procedures are described in Section 4.1.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bensaoud A, Kalita J, Bensaoud M. A survey of malware detection using deep learning. Mach Learn Appl. 2024;16(21):100546. doi:10.1016/j.mlwa.2024.100546.
2. Kim EJ, Lee YK, Lee SM, Kim JN, Kang AR, Kim MS, et al. Malware detection using pre-trained transformer encoder with byte sequences. PLoS One. 2025;20(10):e0332307. doi:10.1371/journal.pone.0332307.
3. Yousuf MI, Anwer I, Riasat A, Zia KT, Kim S. Windows malware detection based on static analysis with multiple features. PeerJ Comput Sci. 2023;9(1):e1319. doi:10.7717/peerj-cs.1319.
4. Chaganti R, Kuppusamy K, Kadry S. A multi-view feature fusion approach for effective malware classification. J King Saud Univ Comput Inf Sci. 2023;35(3):1–12.
5. Imran M, Appice A, Malerba D. Evaluating realistic adversarial attacks against machine learning models for windows PE malware detection. Future Internet. 2024;16(5):168. doi:10.3390/fi16050168.
6. Bao H, Li W, Chen H, Miao H, Wang Q, Tang Z, et al. Stories behind decisions: towards interpretable malware family classification with hierarchical attention. Comput Secur. 2024;144(15):103943. doi:10.1016/j.cose.2024.103943.
7. Yan S, Ren J, Wang W, Sun L, Zhang W, Yu Q. A survey of adversarial attack and defense methods for malware classification in cyber security. IEEE Commun Surv Tutor. 2023;25(1):467–96. doi:10.1109/comst.2022.3225137.
8. Stiborek J, Pevný T, Rehák M. Multiple instance learning for malware classification. Expert Syst Appl. 2018;93(4):346–57. doi:10.1016/j.eswa.2017.10.036.
9. Yang S, Yang Y, Zhao D, Xu L, Li X, Yu F, et al. Dynamic malware detection based on supervised contrastive learning. Comput Electr Eng. 2025;123(4):110108. doi:10.1016/j.compeleceng.2025.110108.
10. Pineau J, Vincent-Lamarre P, Sinha K, Larivière V, Beygelzimer A, d'Alché-Buc F, et al. Improving reproducibility in machine learning research. J Mach Learn Res. 2021;22(164):1–20.
11. Brown A, Gupta M, Abdelsalam M. Automated machine learning for deep learning based malware detection. Comput Secur. 2024;137(1):103582. doi:10.1016/j.cose.2023.103582.
12. Demetrio L, Biggio B, Giacinto G, Roli F. Adversarial attacks on deep-learning-based malware detection. Comput Secur. 2023;123:102917. doi:10.1016/j.cose.2022.102917.
13. Sun Y, Li Z, Wang H, Chen X. MAlign: malware family classification based on raw binary sequence alignment. Comput Secur. 2024;132(11):103379. doi:10.1016/j.cose.2023.103379.
14. Zhang Y, Wang X, Guo P. A multi-view attention-based deep learning framework for malware detection in smart healthcare systems. Comput Commun. 2022;193:206–16. doi:10.1016/j.comcom.2022.07.015.
15. Joyce RJ, Miller G, Roth P, Zak R, Zaresky-Williams E, Anderson H, et al. Ember2024—a benchmark dataset for holistic evaluation of malware classifiers. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2; 2025 Aug 3; New York, NY, USA. p. 5516–26. doi:10.1145/3711896.3737431.
16. Liu Y, Chen H, Zhang X, Wu Q. Contrastive representation learning for Windows malware detection. Sci Rep. 2025;15(1):1–14. doi:10.1038/s41598-025-08556-4.
17. Ganapathiyappan K, Singh JDJ, Loganand S, Jivesh K, Altameem A, Rehman AU, et al. Explainable ensemble learning for robust Android malware detection. Secur Priv. 2026;9(2):e70206. doi:10.1002/spy2.70206.
18. Maniriho P, Mahmood AN, Chowdhury MJ. A systematic literature review on windows malware detection: techniques, research issues, and future directions. J Syst Softw. 2024;209(6):111921. doi:10.1016/j.jss.2023.111921.

19. Sarı NV, Acı M, Acı Çİ. Windows malware detection via enhanced graph representation learning with API call sequences and DLL information. Appl Sci. 2025;15(9):4775. doi:10.13052/2245-1439.741.
20. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Adv Neural Inf Process Syst. 2017;30:5998–6008. doi:10.65215/ctdc8e75.
21. Abuse Ch. MalwareBazaar: malware sample sharing platform. bazaar.abuse.ch [Internet]. 2020 [cited 2026 Feb 18]. Available from: https://bazaar.abuse.ch/.
22. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge, MA, USA: MIT Press; 2016.
23. Kingma DP, Ba JL. Adam: a method for stochastic optimization. arXiv:1412.6980. 2015.
24. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. 2020;585(7825):357–62. doi:10.1038/s41586-020-2649-2.
25. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst. 2019;32:8026–37. doi:10.5555/3454287.3455008.
26. Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. Inf Process Manag. 2009;45(4):427–37. doi:10.1016/j.ipm.2009.03.002.
27. Fawcett T. An introduction to ROC analysis. Pattern Recognit Lett. 2006;27(8):861–74. doi:10.1016/j.patrec.2005.10.010.
28. Li AS, Iyengar A, Kundu A, Bertino E. Revisiting concept drift in windows malware detection: adaptation to real drifted malware with minimal samples. arXiv:2407.13918. 2024.
29. Demetrio L, Biggio B, Lagorio G, Roli F, Armando A. Functionality-preserving black-box optimization of adversarial windows malware. IEEE Trans Inf Forensics Secur. 2021;16:3469–78. doi:10.1109/tifs.2021.3082330.