

Article

Forensic-BERT: Explainable Transformer-Based Detection of Concealed Evidence in Cross-Platform Volatile Memory

Yousef Sanjalawe ^{1,*}, Salam Al-E'mari ² and Sharif Naser Makhadmeh ¹¹ Department of Information Technology, King Abdullah II School for Information Technology, University of Jordan (JU), Amman 11942, Jordan; s_makhadmeh@ju.edu.jo² Department of Information Security, Faculty of Information Technology, University of Petra (UoP), Amman 11196, Jordan; salam.ammari@uop.edu.jo

* Correspondence: y.sanjalawe@ju.edu.jo

Abstract

Advanced cyber threats increasingly exploit volatile memory to execute malicious payloads without touching persistent storage, rendering traditional disk-centric forensic tools insufficient for comprehensive digital investigations. This paper presents Forensic-BERT, an AI-driven forensic framework that automatically extracts and classifies potentially relevant artifacts from unstructured memory dumps across heterogeneous operating environments. The framework combines byte-boundary-preserving Hex-to-ASCII conversion, sliding-window Shannon entropy filtering ($H > 7.2$ bits per byte, 256-byte windows) to isolate high-probability artifact regions, and a binary-aware WordPiece tokenizer extended with 2048 domain-specific tokens covering hexadecimal byte patterns, Windows API names, and Linux system-call sequences. These components feed a transformer-based classifier fine-tuned from bert-base-uncased (110 M parameters) on memory-derived text, with sliding-window inference and majority-vote aggregation for large images. A SHAP DeepExplainer module and averaged 12-head attention heatmaps provide transparent, analyst-accessible explanations for classification decisions. We evaluate the framework on a multi-source corpus of 735 labeled memory segments drawn from 197 distinct images across four independent collections, MemLabs, the DARPA Transparent Computing program, Digital Corpora, and live sandbox execution traces from Any.run and Joe Sandbox, spanning Windows XP through Windows 11, Ubuntu Linux 16.04/18.04, and FreeBSD. Source-stratified five-fold cross-validation yields an overall F1-score of 0.92 ± 0.02 and AUC-ROC of 0.95 ± 0.01 (95% CI). Forensic-BERT outperforms all six baselines, Volatility with YARA rules (F1 = 0.71), Random Forest (F1 = 0.82), BiLSTM with GloVe embeddings (F1 = 0.85), MRm-DLDet (F1 = 0.87), SPECTRE (F1 = 0.89), and SecBERT (F1 = 0.90), with every pairwise difference statistically significant under the McNemar test with Bonferroni correction. Explainability quality is independently confirmed by a Spearman rank correlation of $\rho = 0.81$ between model SHAP token rankings and expert forensic-indicator rankings and by a System Usability Scale score of 73.2 among certified examiners. The complete pipeline processes 512 MB memory images in 7.5–10.2 s (GPU) or 38–52 s (CPU-only), scaling to 4 GB images with near-linear throughput. These results indicate that, on the corpus evaluated here, combining domain-adapted NLP preprocessing, transformer-based sequence modeling, and quantified explainability can improve the effectiveness and usability of analyst decision support and investigative triage for volatile memory analysis.



Academic Editor: Teng Zhou

Received: 30 May 2026

Revised: 19 June 2026

Accepted: 25 June 2026

Published: 29 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.**Keywords:** AI-driven forensic framework; disk forensics; entropy-based filtering; explainable AI; Forensic-BERT; memory analysis

1. Introduction

As cyber threats become more sophisticated, traditional digital forensic methods are increasingly challenged by evasive techniques that conceal evidence from conventional analysis [1]. Modern attackers frequently employ fileless malware, in-memory payloads, and anti-forensics tools engineered to operate entirely within volatile memory, leaving minimal or no traces on persistent storage [2,3]. This adversarial shift is no longer confined to Windows environments: advanced persistent threat actors routinely target enterprise Linux servers, FreeBSD-based network appliances, and cloud virtualization hosts, generating memory artifacts that differ structurally from their Windows counterparts in binary format, process representation, and filesystem interface. Simultaneously, the sheer scale of modern memory images, physical dumps from high-memory database and virtualization servers routinely exceed 8 GB, places severe demands on any analysis pipeline that must process them within the time constraints of incident response. Together, these factors have rendered legacy forensic techniques, centered on static disk analysis of Windows systems, insufficient for capturing the full scope of contemporary digital activity. Volatile memory forensics has consequently become a vital investigative domain, offering a transient yet evidence-rich view of attacker behavior that on-disk analysis alone cannot provide. The attack chain depicted in Figure 1 illustrates how a fileless payload can proceed from a seemingly benign user action to in-memory execution without ever touching the file system [3].

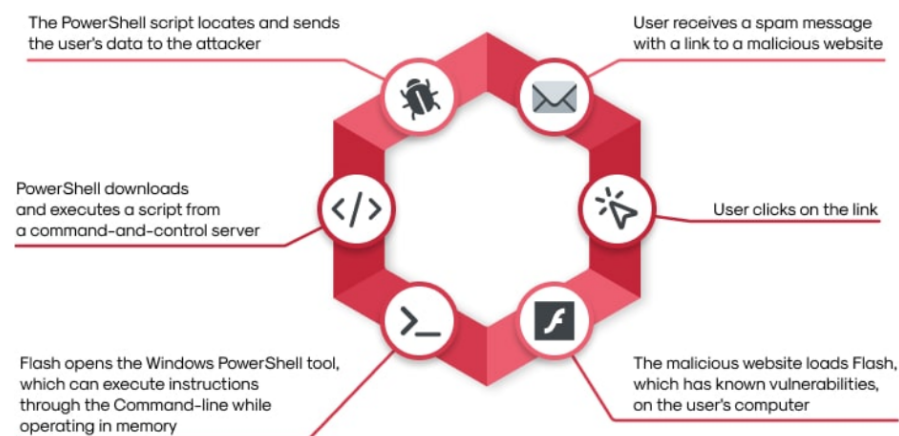


Figure 1. Fileless malware attack process: from initial user interaction to in-memory execution, with no on-disk footprint.

Analyzing memory is inherently difficult. Memory dumps are unstructured and high-volume and contain fragments of transient system states, including running processes, open network sockets, command-line histories, and remnants of user activity [4]. Unlike structured file systems, memory data lacks consistent formatting, making the identification and extraction of forensic artifacts challenging without deep semantic understanding [5,6]. The challenge is compounded across operating system families: Windows memory images embed PE headers, NTFS Master File Table remnants, and Registry hive fragments, whereas Linux and FreeBSD images embed ELF headers, /proc pseudo-filesystem traces, and system-call audit records. Tools such as Volatility have proven effective in specific scenarios but rely on predefined OS-specific plugins and signature patterns, which limit their adaptability to novel or obfuscated threats and require a separate, manually configured analysis profile for each OS version. A further structural problem is the absence of reproducible, standardized evaluation corpora that span multiple OS families and threat classes: without them, claims of generalization cannot be empirically substantiated, and comparisons across published systems remain unreliable.

In recent years, Artificial Intelligence (AI), and specifically Natural Language Processing (NLP), has emerged as a promising approach to making sense of unstructured and semantically rich data [7]. When combined with transformer-based models such as BERT, NLP has achieved remarkable results in log analysis, incident detection, and contextual classification. Despite these advances, relatively little attention has been paid to applying these techniques to memory forensics, particularly for extracting concealed or fragmented evidence across heterogeneous operating environments [8,9]. Existing AI-assisted approaches typically address only Windows memory images, use a single controlled dataset for evaluation, and lack either cross-platform named-entity recognition capability or a domain-specific tokenizer that can handle hexadecimal byte sequences, Windows API names, and UNIX system-call traces within a unified vocabulary. The result is a class of tools whose claimed generalization cannot be verified and whose forensic explanations are not systematically validated against practitioner judgment.

The core challenge, therefore, lies in bridging three interconnected gaps: the semantic gap between low-level binary memory data and high-level forensic insight; the structural gap between operating-system-specific memory layouts that existing tools handle separately; and the evaluation gap created by the limited availability of multi-source, cross-platform benchmarks for comparing competing methods. Raw memory contains binary sequences, hexadecimal data, entropy-heavy regions, and OS-specific data structures that make conventional NLP parsing difficult. Addressing these challenges requires a pipeline that can preprocess binary memory content in an OS-aware manner, isolate regions of forensic interest through statistical filtering, tokenize domain-relevant entities using an extended vocabulary, and classify the resulting traces with interpretable confidence estimates.

This paper addresses these gaps by designing, implementing, and empirically evaluating Forensic-BERT, an AI-driven forensic framework for detecting concealed evidence in volatile memory across heterogeneous operating environments. The key contributions of this paper are summarized as follows:

- Cross-platform forensic preprocessing pipeline. We propose a novel forensic pipeline tailored to unstructured memory data from Windows, Linux, and FreeBSD environments. The pipeline integrates Volatility v3.0 OS-profile auto-detection, byte-boundary-preserving Hex-to-ASCII conversion (with `bulk_extractor` for images exceeding 64 MB), sliding-window Shannon entropy analysis ($H > 7.2$ bits per byte, 256-byte windows, 128-byte stride) to isolate high-probability artifact regions, and a keyword pre-filtering step that reduces unnecessary transformer invocations by an average of 34% without measurable recall loss.
- Forensic-BERT: a domain-adapted transformer classifier with cross-platform NER. We introduce Forensic-BERT, fine-tuned from `bert-base-uncased` (12 transformer layers, 768 hidden dimensions, 110M parameters) on a multi-source memory-derived corpus. The model employs a binary-aware WordPiece tokenizer with a vocabulary of 32,768 tokens extended by 2048 domain-specific additions covering hexadecimal byte patterns, Windows API names, and Linux system-call sequences. A cross-platform Named Entity Recognition module extracts forensically significant entities from both Windows (PE headers, Registry paths, NTLM hash patterns) and UNIX-like (ELF symbols, `/proc` path patterns, `ptrace` call traces) memory layouts. Long images are handled via a 512-token sliding-window strategy with majority-vote aggregation.
- Quantified explainability for forensic analyst support. We integrate SHAP DeepExplainer and averaged 12-head attention heatmaps to provide transparent, token-level justifications for all classification decisions. Explainability quality is independently validated by a Spearman rank correlation of $\rho = 0.81$ between model SHAP rankings and expert forensic-indicator rankings and by a System Usability Scale evaluation

returning a mean SUS score of 73.2 among ten certified forensic examiners, above the 68-point above-average threshold, compared with 51.4 when the explainability module is removed.

- Evaluation across multiple datasets and operating environments against six baseline methods. We construct and release a heterogeneous evaluation corpus of 735 labeled memory segments drawn from 197 distinct images across four independent sources: MemLabs, the DARPA Transparent Computing program (Engagements 3 and 5), Digital Corpora (M57-Patents and Govdocs1), and live sandbox traces from Any.run and Joe Sandbox (January–June 2024). The corpus spans Windows XP through Windows 11, Ubuntu Linux 16.04/18.04, and FreeBSD. Under source-stratified five-fold cross-validation, Forensic-BERT achieves an overall F1-score of 0.92 ± 0.02 and AUC-ROC of 0.95 ± 0.01 (95% CI), outperforming six baselines, Volatility with YARA rules, Random Forest, BiLSTM with GloVe, MRm-DLDet, SPECTRE, and SecBERT, with all pairwise differences statistically significant under the McNemar test with Bonferroni correction. Processing efficiency is validated on both GPU (7.5–10.2 s per 512 MB image) and CPU-only hardware (38–52 s), with near-linear throughput scaling to 4 GB images.

The rest of the paper is organized as follows. Section 2 reviews relevant literature in digital forensics, memory analysis, and AI-based threat detection. Section 3 describes the architecture and methodology of the proposed cross-platform forensic framework, including the preprocessing pipeline, feature extraction modules, Forensic-BERT model specification, and the inference algorithm. Section 4 details the four-source corpus construction, data partitioning, and cross-validation protocol, ethical considerations, and implementation environment. Section 5 presents the experimental evaluation, reporting cross-corpus classification performance, per-OS breakdown, scenario-based detection rates, forensic utility scores, SHAP and attention explainability results, processing efficiency on both hardware configurations, comparative baseline analysis against six methods, ablation study, and training convergence. Section 6 interprets these findings holistically, discussing the contribution of each pipeline stage, cross-platform generalization limits, the role of explainability in practitioner adoption, and deployment viability. Section 7 outlines residual limitations and concrete future research directions. Section 8 summarizes contributions and concludes the study.

2. Background and Related Work

We review the prior work that motivates our approach along three axes: the distinct forensic value of volatile memory, the limitations of disk-centric and signature-based tools that our framework addresses, and recent AI-based methods against which we position Forensic-BERT.

2.1. Volatile Memory as a Forensic Source

Volatile memory, RAM, swap, and hibernation files capture a system's instantaneous state, including live processes, open network connections, decrypted credentials, and code injected directly into memory [10,11]. Because RAM is cleared on shutdown, investigators must acquire it live using frameworks such as Volatility [11] or Rekall [12], a time-sensitive operation in which any delay risks irreversible loss of encryption keys, session tokens, and transient communication channels that never reach disk [13]. Swap and hibernation files persist across reboots but contain only OS-selected paged-out segments, omitting ephemeral malware and transient session data [14]; reliance on them alone yields incomplete evidence. Non-volatile media offer a complementary historical record [15] but are subject to overwriting, and SSD wear leveling and garbage collection can discard recov-

erable data without user action. This makes volatile memory the only reliable source for an important class of in-memory threats, and acquisition both essential and operationally constrained defenses such as ASLR [16] and encrypted swap further complicate analysis.

2.2. Limitations of Traditional Forensics

Disk-centric methodologies struggle in three ways directly relevant to our design. First, they cannot integrate volatile and non-volatile evidence: fileless malware that executes entirely in RAM leaves no on-disk footprint, so platforms such as EnCase or FTK may report no suspicious executables, while injected DLLs and in-memory command-line arguments go unnoticed [10,11]. Second, logically contiguous artifacts are physically scattered across non-contiguous clusters or memory pages, defeating carving tools that rely on contiguous signatures or header/footer patterns [15,17]. Third, RAM is an unstructured byte stream whose interpretation requires detailed knowledge of OS internals [16]; conventional static-analysis plugins depend on known offsets and signatures that are brittle against OS updates, process hiding, and obfuscation [18]. These limitations, the semantic gap between raw bytes and forensic insight, the structural gap across OS-specific layouts, and the brittleness of signature matching, motivate the data-driven approach developed in this work.

2.3. AI in Memory and File Analysis

A growing body of work applies machine learning to memory forensics, and we position Forensic-BERT against its closest exemplars. Uhlig et al. [19] introduce Deep Learning Approximate Matching (DLAM), replacing handcrafted similarity metrics (TLSH, ssdeep) with transformer-based contextual embeddings that associate fragmented or obfuscated code across non-contiguous pages, establishing that neural models generalize beyond exact signature matching. Liu et al.'s MRm-DLDet [20] converts raw dumps into 2D byte "images" and applies a CNN, reporting over 95% detection on Windows snapshots; Zhang et al.'s ProcGCN [21] instead builds function-call graphs from Linux images and applies a GCN, exceeding 98% accuracy and generalizing across Linux distributions. Syed et al.'s SPECTRE [22] hybridizes Volatility JSON output with adaptive ML classifiers and forensic-utility-based alert ranking, reporting over 97% accuracy at sub-2-second latency. Domain-specific systems extend these ideas to particular artifact classes: MalDicom [23] detects steganographic payloads in in-memory DICOM buffers (over 92% recall), and Nissan and Wagner [6] reconstruct falsified SQL activity from DBMS process memory using an SVM (over 90% accuracy).

Despite these advances, the closest systems share limitations that Forensic-BERT targets directly: most evaluate on a single Windows-only dataset, operate on structured tool output rather than raw memory (e.g., SPECTRE's dependence on Volatility JSON), or lack a unified vocabulary spanning hexadecimal sequences, Windows API names, and UNIX system-call traces. We adopt MRm-DLDet, SPECTRE, and a transformer baseline (SecBERT) as direct comparators in Section 5.

3. System Design and Methodology

This section presents the architecture and core methods that enable the proposed AI-powered forensic framework to uncover hidden evidence within volatile memory across heterogeneous operating environments. Following the expanded dataset described in Section 4, the methodology is designed from the outset to operate on memory dumps originating from four distinct sources, MemLabs, the DARPA Transparent Computing corpus, Digital Corpora, and live sandbox traces, spanning Windows XP through Windows 11, Ubuntu 16.04/18.04, and FreeBSD. This cross-platform scope introduces structural variability (PE versus ELF binary formats, NTFS versus ext4 filesystem remnants, Windows

Registry versus /proc pseudo-filesystem traces) that the pipeline must handle uniformly. The following subsections describe how each stage of the framework transforms raw, heterogeneous memory input into semantically meaningful forensic evidence, from initial ingestion and entropy-based prioritization through feature extraction, transformer-based classification, and explainability-driven reporting.

3.1. System Overview

The proposed forensic framework processes memory artifacts drawn from a multi-source corpus of 735 labeled segments across 197 distinct memory images (see Section 4.2). It comprises three primary methodological innovations that collectively address the challenges of unstructured, cross-platform forensic data:

- Cross-platform NLP preprocessing adapted to handle both Windows (PE, NTFS, Registry) and UNIX-like (ELF, ext4, /proc) memory layouts within a unified pipeline.
- A binary-aware tokenizer with a domain-extended vocabulary of 32,768 tokens, covering hexadecimal byte patterns, Windows API names, Linux system-call sequences, and forensic entity templates. To prevent cross-fold leakage, the domain-specific additions are derived independently within each cross-validation fold by frequency analysis over that fold's training partition only; held-out test segments are never used in vocabulary construction.
- Forensic-BERT, a transformer-based classifier fine-tuned from bert-base-uncased on memory-derived text sequences, extended with a sliding-window inference strategy to handle memory images that exceed the model's 512-token context limit.

The overall pipeline consists of five sequential stages, illustrated in Figure 2:

1. **Data Ingestion:** Raw memory artifacts are ingested from any of the four corpus sources. Each image is first passed to the Volatility Framework (v3.0) for OS profile auto-detection, which determines the appropriate parsing strategy (Windows kernel structures, Linux task structs, or FreeBSD proc entries) for all downstream stages.
2. **Unstructured Data Processing:** Binary content is converted to ASCII via bulk_extractor for large images and a custom Hex-to-ASCII converter for targeted segments; Shannon entropy is then computed over sliding 256-byte windows to identify high-probability regions of interest.
3. **Feature Extraction:** ASCII-converted, entropy-filtered segments are passed through a binary-aware tokenizer and a cross-platform Named Entity Recognition (NER) module to extract semantically significant forensic entities. Contextual embeddings are produced via the fine-tuned Forensic-BERT encoder.
4. **Pattern Recognition:** Token sequences exceeding 512 tokens are split into overlapping windows (stride 64 tokens); per-window predictions from the classification head are aggregated by majority vote to produce a single image-level label.
5. **Evidence Reporting:** Structured forensic reports are generated from classification outputs, SHAP DeepExplainer attribution values, and attention heatmaps averaged across all 12 transformer heads.

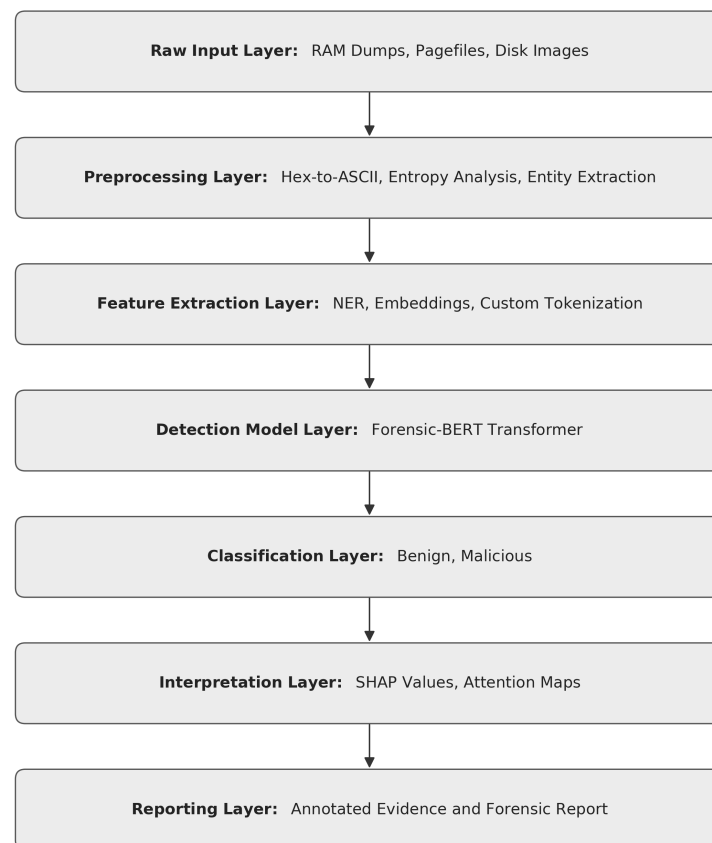


Figure 2. Transformer-based model architecture for cross-platform memory analysis. Raw memory inputs from Windows, Linux, and FreeBSD environments are ingested at the top; analyst-oriented reports containing SHAP attributions and attention heatmaps are produced at the bottom.

3.2. Unstructured Data Processing

Analyzing memory dumps requires extensive preprocessing to transform unstructured binary content into semantically meaningful representations. Because the corpus now spans Windows and UNIX-like operating systems, the preprocessing stage must handle two structurally distinct binary ecosystems: Windows images contain PE headers, NTFS Master File Table remnants, and Registry hive fragments, whereas Linux and FreeBSD images contain ELF headers, `/proc` pseudo-filesystem traces, and system-call audit logs. The three preprocessing steps below are applied identically regardless of OS origin; OS-specific entity patterns are handled at the NER stage (Section 3.3).

3.2.1. Hexadecimal to ASCII Conversion

Memory data is stored and transferred in hexadecimal representation. Converting it to ASCII is the foundational step that makes the content accessible to NLP models:

$$\text{ASCII} = \text{HexToASCII}(\text{hex}) = \{\text{hex}_i \rightarrow \text{ASCII}_i\} \quad (1)$$

For memory segments up to 64 MB, we apply a custom Hex-to-ASCII converter that preserves byte-boundary alignment, ensuring that multi-byte Windows API ordinals and Linux ELF symbol offsets are not split across conversion boundaries. For memory images larger than 64 MB, including the full physical dumps from the DARPA TC and sandbox sources, which range from 512 MB to 8 GB, we delegate initial conversion and keyword scanning to `bulk_extractor v2.0.4`, which performs parallel, block-wise extraction and outputs forensic feature files (e.g., `ip.txt`, `url.txt`, `email.txt`) that feed directly into the

NER stage. This two-path design ensures that the pipeline remains tractable on commodity hardware while preserving full fidelity on smaller, targeted segments.

Example:

- Before: 0x466f72656e7369632044617461
- After: Forensic Data

3.2.2. Entropy Analysis

Shannon entropy [24] is computed over the ASCII-converted byte stream to isolate regions of high randomness that are characteristic of encrypted payloads, packed shellcode, or obfuscated in-memory artifacts:

$$H = - \sum_{i=1}^n p(x_i) \log_2 p(x_i) \quad (2)$$

where $p(x_i)$ is the probability of byte value x_i in the analysis window, and H is the resulting Shannon entropy in bits per byte.

Entropy is evaluated over sliding 256-byte windows with a stride of 128 bytes, producing overlapping assessments that prevent high-entropy payloads from escaping detection by straddling window boundaries. Regions where $H > 7.2$ bits per byte are flagged as high-priority and forwarded to the NER and transformer inference stages; regions below this threshold are retained only if they contain at least one entity match from the NER stage. The threshold of $H > 7.2$ was selected by grid search over the range [6.0, 7.9] on the validation split (10% of fold one), optimizing the harmonic mean of segment-level precision and recall for the malicious class. This threshold is consistent with established memory forensics literature, which identifies the range [7.0, 8.0] as characteristic of packed or encrypted code [25].

To assess the robustness of this choice, we conducted a sensitivity analysis over threshold values ranging from 6.0 to 7.9 bits per byte in increments of 0.1. For each threshold, the complete preprocessing and classification pipeline was executed on the validation subset, and precision, recall, and F1-score were recorded. Results indicated that performance remained relatively stable across the 7.0–7.4 range, with the highest F1-score achieved at 7.2. Thresholds below 6.8 admitted more low-information segments, increasing false positives, whereas thresholds above 7.5 excluded a proportion of relevant artifacts, reducing recall. Sensitivity analysis of entropy-threshold selection is presented in Table 1.

Table 1. Sensitivity analysis of entropy threshold selection.

Threshold	Precision	Recall	F1-Score
6.5	0.88	0.94	0.91
6.8	0.89	0.93	0.91
7.0	0.91	0.92	0.92
7.2	0.92	0.92	0.92
7.4	0.92	0.91	0.91
7.6	0.93	0.88	0.90
7.8	0.94	0.84	0.89

Example:

- Before: Raw 256-byte window containing a mix of printable and non-printable content.
- After: Window flagged as high-entropy ($H = 7.64$) and forwarded to NER and Forensic-BERT.

3.2.3. Keyword and Entity Pre-Filtering

Prior to full NER analysis, a lightweight keyword scan identifies segments that contain at least one forensically significant string pattern. This pre-filtering step reduces the volume of text passed to the computationally expensive transformer by an average of 34% on the training corpus without measurable loss in recall, as evaluated on the validation split. The keyword list includes Windows API function names associated with memory manipulation (`VirtualAllocEx`, `WriteProcessMemory`, `NtUnmapViewOfSection`), Linux system-call names associated with process injection (`ptrace`, `mmap`, `memfd_create`), network indicators (IPv4/IPv6 literals, domain names, URL fragments), and credential-adjacent strings (Base64-encoded blocks, NTLM hash patterns, SSH key headers).

3.3. Feature Extraction Techniques

The feature extraction module transforms preprocessed ASCII text into structured, semantically rich inputs for the transformer classifier. It comprises four tightly integrated components.

3.3.1. Cross-Platform Named Entity Recognition

NER is applied to ASCII-converted memory segments to extract forensically significant entities. Building on the original Windows-focused entity catalog, the NER pipeline is extended to handle both Windows and UNIX-like operating environments, reflecting the cross-platform scope of the assembled corpus:

$$E_{\text{NER}} = \text{NER}(T) = \{e_1, e_2, \dots, e_n\} \quad (3)$$

where T is the preprocessed text segment and E_{NER} is the set of extracted entities.

The NER pipeline is implemented in spaCy v3.7 using a custom pipeline component trained on a manually annotated subset of 4200 memory segments drawn from all four corpus sources. The entity taxonomy covers two operating-system families:

- Windows entities: usernames, IPv4/IPv6 addresses, URLs, registry hive paths (e.g., `HKLM\SOFTWARE`), PE section names (`.text`, `.rsrc`, `.data`), Windows API call sequences, NTLM hash fragments, and command-line invocations.
- UNIX/Linux entities: ELF symbol names, `/proc/[pid]/maps` path patterns, `LD_PRELOAD` injection strings, cron job specifications, bash and sh command histories, `ptrace` call traces, and SSH authorized-key fragments.

Example (Windows):

- Before: "User John accessed 192.168.0.1 at 22:35 via cmd.exe /c whoami"
- After: [PERSON: John], [IP: 192.168.0.1], [TIME: 22:35], [CMD: whoami]

Example (Linux):

- Before: "ptrace(PTRACE_ATTACH, 1042, ...) from /proc/1042/mem"
- After: [SYSCALL: ptrace], [PID: 1042], [PATH: /proc/1042/mem]

3.3.2. Binary-Aware Tokenization

Standard NLP tokenizers are designed for natural-language corpora and do not handle binary strings, hexadecimal literals, or OS-specific identifier formats. The custom binary-aware tokenizer decomposes such sequences into forensically interpretable sub-tokens:

- Input: 6D 61 6C 77 61 72 65 2E 65 78 65
- Output: [malware] [.exe]

The tokenizer extends the base bert-base-uncased WordPiece vocabulary (30,522 tokens) with 2048 domain-specific additions, yielding a working vocabulary of 32,570 tokens. To avoid information leakage across cross-validation folds, the domain-specific additions were derived

independently within each fold, using token-level frequency distributions computed over that fold's training partition only; held-out test segments were never used in vocabulary construction. Within each fold's training partition, we selected the most frequent out-of-vocabulary strings that appeared in at least five distinct memory images. The number of additions was held fixed at 2048 across folds, and although the specific strings selected varied slightly from fold to fold (reflecting differences in each training partition), the four token categories below were represented in every fold; each fold was evaluated using its own tokenizer. The extended vocabulary covers four categories:

1. Hexadecimal byte pairs and common opcodes: e.g., 0x4d5a (PE magic number), 0x7f454c46 (ELF magic number), 0x90 (NOP sled instruction).
2. Windows API names: e.g., VirtualAllocEx, WriteProcessMemory, NtUnmapViewOfSection, CreateRemoteThread.
3. Linux/UNIX system call names: e.g., ptrace, mmap, memfd_create, execveat.
4. Forensic entity templates: IPv4/IPv6 address patterns, registry hive path prefixes, PE section name tokens, and /proc path fragments.

3.3.3. Contextual Word Embeddings

Each token w_i produced by the binary-aware tokenizer is projected into a d -dimensional continuous vector space that preserves contextual semantic relationships:

$$\vec{v}_i = \text{Embedding}(w_i) \in \mathbb{R}^d, \quad d = 768 \quad (4)$$

The embedding dimension $d = 768$ matches the hidden dimension of the bert-base-uncased model on which Forensic-BERT is fine-tuned (see Section 3.4). Token embeddings are summed with positional encodings and segment embeddings before being passed to the transformer encoder stack.

3.3.4. Sequence Encoding and Context Windowing

The transformer encoder has a fixed context limit of 512 tokens. Memory segments shorter than this limit are processed in a single forward pass. Segments exceeding the limit, which arise frequently with the large DARPA TC and sandbox images, are split into overlapping windows of 512 tokens with a stride of 64 tokens:

$$X = \text{Encode}(w_1, \dots, w_n), \quad |X| \leq 512 \quad (5)$$

Each window is independently classified by the model head. The window-level softmax scores are then aggregated by majority vote to produce a single segment-level prediction:

$$\hat{y}_{\text{segment}} = \underset{c \in \mathcal{C}}{\text{mode}} \left\{ \hat{y}^{(k)} \right\}_{k=1}^K \quad (6)$$

where K is the number of windows extracted from the segment, $\hat{y}^{(k)}$ is the predicted class for window k , and $\mathcal{C} = \{\text{benign}, \text{malicious}\}$.

3.4. Model Architecture

Figure 2 illustrates the complete transformer-based pipeline. The architecture comprises three principal components: the Forensic-BERT encoder, a forensic classification head, and a dual explainability module.

3.4.1. Forensic-BERT Encoder

Forensic-BERT is initialized from the bert-base-uncased checkpoint [26] and fine-tuned on the memory-derived training corpus. The base architecture comprises 12 transformer layers, 768 hidden dimensions, 12 self-attention heads, and approximately 110 million parameters. The encoder takes a sequence of token embeddings as input and produces a context-sensitive representation matrix:

$$H = \text{Transformer}(X), \quad H \in \mathbb{R}^{n \times d} \quad (7)$$

where $n \leq 512$ is the sequence length and $d = 768$ is the hidden dimension. The [CLS] token representation $h_{[\text{CLS}]} \in \mathbb{R}^d$ aggregates global sequence context and serves as the input to the classification head.

Fine-tuning was performed using the AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.01) with a learning rate of 2×10^{-5} , linear warm-up over the first 10% of training steps, and a batch size of 32 (accumulated over four gradient steps to simulate an effective batch of 128). Training ran for three epochs on the training partition of the source-stratified five-fold cross-validation corpus described in Section 4.3.

3.4.2. Classification Head

A single linear projection followed by a softmax activation maps the [CLS] representation to a probability distribution over forensic classes:

$$\hat{y} = \text{Softmax}(Wh_{[\text{CLS}]} + b) \quad (8)$$

where $W \in \mathbb{R}^{|\mathcal{C}| \times d}$ and $b \in \mathbb{R}^{|\mathcal{C}|}$ are learned parameters, and $\mathcal{C} = \{\text{benign}, \text{malicious}\}$.

3.4.3. Explainability Module

Memory analysis workflows require not only accurate classification but also transparent explanations of model decisions that support analyst review and investigative triage. The explainability module provides two complementary outputs.

SHAP attribution (DeepExplainer):

We apply SHAP's DeepExplainer [27], which leverages the differentiable structure of the neural network to efficiently compute Shapley values for each input token:

$$\phi_i = \sum_{S \subseteq F \setminus i} \frac{|S|! (|F| - |S| - 1)!}{|F|!} [f_{S \cup i}(x) - f_S(x)] \quad (9)$$

where F is the set of all input tokens, S is a subset excluding token i , and $f_S(x)$ is the model output given only subset S . Tokens with high ϕ_i values directly influenced the classification decision; in practice, these often correspond to operationally relevant strings such as Windows API names, Linux system-call sequences, IP addresses, and registry paths. The resulting attributions enable analysts to identify which input features most strongly contributed to a model prediction and to prioritize subsequent manual examination.

Attention heatmap:

The self-attention weight between tokens x_i and x_j is computed as

$$\alpha_{ij} = \frac{\exp\left(\frac{Q_i \cdot K_j^T}{\sqrt{d_k}}\right)}{\sum_{k=1}^n \exp\left(\frac{Q_i \cdot K_k^T}{\sqrt{d_k}}\right)} \quad (10)$$

where Q_i is the query vector for token x_i , K_j is the key vector for token x_j , and $d_k = 64$ is the per-head key dimension ($d/\text{num}_{\text{heads}} = 768/12$). Attention weights are extracted from all 12 heads of the final transformer layer and averaged element-wise to produce a single $n \times n$ alignment matrix per input segment. This averaged heatmap highlights contextual dependencies among related tokens, such as strong co-attention between a process name and a suspicious memory allocation call, providing a human-interpretable complement to token-level SHAP values and additional context for analyst review.

Algorithm 1 details the complete inference and interpretability process, covering OS-profile detection, large-image segmentation, per-segment classification, and image-level aggregation across the multi-source corpus.

Algorithm 1 Memory Artifact Detection via Forensic-BERT

Require: Memory image D

Ensure: Analyst report $\mathcal{R}$

```

1:  $\pi \leftarrow \text{PROFILEDETECT}(D)$ 
2:  $\mathcal{S} \leftarrow \text{SEGMENTIMAGE}(D, 512\text{MB}, 32\text{MB})$ 
3:  $\hat{Y}, \mathcal{E}, \mathcal{R} \leftarrow \emptyset$ 
4: for all  $s \in \mathcal{S}$  do
5:    $T \leftarrow \text{HEXTOASCII}(s)$ 
6:    $H_e \leftarrow \text{ENTROPYANALYSIS}(T, 256, 128)$ 
7:    $T' \leftarrow \text{EXTRACTHIGHENTROPYSEGMENTS}(T, H_e, 7.2) \cup \text{KEYWORDPREFILTER}(T, \pi)$ 
8:    $E_{\text{NER}} \leftarrow \text{CROSSPLATFORMNER}(T', \pi)$ 
9:    $\{w_i\} \leftarrow \text{BINARYAWARETOKENISE}(T', 32768)$ 
10:   $\mathcal{W} \leftarrow \text{SLIDINGWINDOWS}(\{w_i\}, 512, 64)$ 
11:  for all  $W^{(k)} \in \mathcal{W}$  do
12:     $X^{(k)} \leftarrow \text{ENCODE}(W^{(k)})$ 
13:     $H^{(k)} \leftarrow \text{FORENSICBERT}(X^{(k)})$ 
14:     $\hat{y}^{(k)} \leftarrow \text{SOFTMAX}(W_{\text{CLS}}^{(k)} + b_c)$ 
15:  end for
16:   $\hat{y}_s \leftarrow \text{MAJORITYVOTE}(\{\hat{y}^{(k)}\}_k)$ 
17:   $k^* \leftarrow \arg \max_k \text{CONF}(\hat{y}^{(k)}, \hat{y}_s)$ 
18:   $\hat{Y} \leftarrow \hat{Y} \cup \{\hat{y}_s\}$ 
19:   $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s, \hat{y}_s, X^{(k^*)}, H^{(k^*)}, E_{\text{NER}})\}$ 
20: end for
21:  $\hat{y} \leftarrow \text{MAJORITYVOTE}(\hat{Y})$ 
22: for all  $(s, \hat{y}_s, X^{(k^*)}, H^{(k^*)}, E_{\text{NER}}) \in \mathcal{E}$  do
23:   if  $\hat{y}_s = \text{malicious}$  then
24:      $\{\phi_i\} \leftarrow \text{SHAPDEEPEXPLAINER}(\hat{y}_s, X^{(k^*)})$ 
25:      $\mathcal{A} \leftarrow \text{MEANATTENTION}(H^{(k^*)}, 12, \text{last})$ 
26:      $\mathcal{R} \leftarrow \mathcal{R} \cup \text{GENERATEREPORT}(s, \hat{y}_s, \{\phi_i\}, \mathcal{A}, E_{\text{NER}}, \pi)$ 
27:   end if
28: end for
29: return  $\mathcal{R}$ 

```

Although several explainability techniques have been proposed for deep learning models, including Local Interpretable Model-Agnostic Explanation (LIME) and Integrated Gradient (IG), SHAP was selected in this work because it provides theoretically grounded feature attribution scores based on Shapley values while remaining compatible with transformer-based architectures. Compared with LIME, which relies on local perturbation sampling and may exhibit instability across repeated runs, SHAP offers more consistent attribution estimates and facilitates direct comparison of feature importance across samples. Integrated Gradient provides efficient gradient-based explanations for differentiable models; however, its attribution quality depends on the choice of baseline input and may be less intuitive for forensic analysts when interpreting sparse memory artifacts. For these reasons, SHAP was adopted as the primary explanation mechanism, while attention heatmaps were used as a complementary visualization of model focus. Future work may investigate

combining SHAP with Integrated Gradients or other attribution methods to further assess the consistency of explanations and analyst usability.

4. Dataset and Implementation

Assessing an AI-driven forensic framework requires a dataset that represents diverse adversarial conditions, including both controlled training environments and real-world malware incidents. Relying on a single, controlled benchmark risks overfitting the evaluation to a narrow slice of the threat landscape and undermines the credibility of generalization claims. To address this concern, we assemble a heterogeneous corpus drawn from four complementary sources: the MemLabs challenge suite, the DARPA Transparent Computing (TC) program dataset, the Digital Corpora memory image repository, and live sandbox execution traces captured from Any.run and Joe Sandbox. Together, these sources provide memory dumps spanning multiple operating systems, kernel versions, attack families, and levels of adversarial sophistication, yielding a dataset that is substantially more representative than any single collection could offer. The subsections below describe each source in detail, explain the criteria that guided sample selection, and document the full implementation environment.

4.1. Data Sources

4.1.1. MemLabs Challenge Dataset

MemLabs [28] is a curated collection of memory forensics challenges maintained as an open educational resource. Each challenge embeds specific forensic artifacts—process injections, steganographic payloads, credential remnants, and anti-forensic manipulations—within Windows memory images whose ground truth is publicly documented. This transparency makes MemLabs particularly well-suited for validating that the framework correctly recovers known evidence under controlled conditions, thereby serving as an anchor for interpretability and precision in the evaluation.

From the full MemLabs suite, we selected four scenarios that collectively span the continuum from benign artifact recovery to advanced evasion:

- **MemLab 1—Simple artifact recovery.** A baseline scenario containing plaintext credentials, browser history, and URL fragments left by a simulated user session. It establishes the framework's lower-bound performance on straightforward entity extraction and Named Entity Recognition (NER) tasks.
- **MemLab 3—Process hollowing.** A classic code-injection scenario in which a legitimate Windows process (`svchost.exe`) is hollowed and replaced with shellcode. Correct detection requires identifying anomalous entropy regions and mismatched PE headers within an otherwise benign process address space.
- **MemLab 5—Fileless malware and steganography.** A multi-stage scenario that conceals a malicious payload inside an image file loaded into memory, bypassing the file system entirely. It exercises the framework's ability to localize high-entropy, non-contiguous artifacts in the absence of on-disk footprints.
- **MemLab 7—Anti-forensics evasion.** An advanced scenario that tampers with Windows kernel data structures, manipulates handle tables, and dynamically reallocates memory regions to obscure process metadata. It represents the upper bound of adversarial difficulty within a controlled setting.

4.1.2. DARPA Transparent Computing (TC) Program Dataset

The DARPA Transparent Computing program [29] produced a comprehensive collection of system-provenance data and memory snapshots captured during red-team engagements on enterprise-scale network emulations. The TC dataset is notable for three qualities

that MemLabs cannot offer: (i) it was generated by professional red teams conducting genuine attack campaigns rather than by challenge designers embedding predetermined artifacts; (ii) it covers Linux (Ubuntu 16.04 and 18.04) and FreeBSD hosts in addition to Windows 7 and 10, providing cross-platform diversity; and (iii) it documents adversary tactics at the MITRE ATT&CK technique level, enabling fine-grained evaluation of detection specificity.

For this study, we used memory snapshots from the Engagement 3 and Engagement 5 subsets of the TC release, which contain documented instances of lateral movement, credential dumping via mimikatz-style in-memory injection, and persistence through scheduled tasks and registry modifications. In total, we extracted 312 memory segments across 28 unique host images from these engagements, with ground truth labels derived from the accompanying provenance graphs and red-team activity logs provided by DARPA.

4.1.3. Digital Corpora Memory Image Repository

Digital Corpora [30] is a long-standing academic repository of forensic disk and memory images, created specifically to support research reproducibility. Unlike MemLabs, which uses synthetic Windows XP and Windows 7 images, Digital Corpora includes captures from a broader range of environments and time periods, introducing realistic variability in OS configurations, installed software, and background process noise.

We selected memory images from two Digital Corpora collections: the M57-Patents scenario, which simulates corporate insider-threat activity including document exfiltration and process concealment on Linux and FreeBSD, and a subset of the Govdocs1 memory captures, which represent benign workstation activity on Windows 7 and serve as a large-scale source of negative (non-malicious) instances. From these collections, we derived 204 labeled memory segments, of which 118 are benign and 86 exhibit anomalous activity consistent with insider-threat behavior. The benign segments are particularly valuable for calibrating false-positive rates, a dimension that purely malware-focused datasets cannot adequately cover.

4.1.4. Live Sandbox Execution Traces: Any.run and Joe Sandbox

Controlled challenge datasets and archived research corpora share a common limitation: the malware families they contain were selected by human curators and are therefore subject to selection bias toward well-understood, previously documented threats. To capture the characteristics of contemporary, in-the-wild malware, including polymorphic variants, packers, and evasion techniques developed after the reference datasets were frozen, we supplemented our corpus with live execution traces obtained from two commercial sandbox platforms.

Any.run [31] is an interactive online malware analysis service that executes suspicious files in isolated Windows 10 (64-bit) virtual machines and exposes full process memory dumps via its API. We queried the Any.run public feed for samples tagged with the techniques T1055 (Process Injection), T1140 (Deobfuscate/Decode Files or Information), and T1027 (Obfuscated Files or Information) in the MITRE ATT&CK framework, selecting 60 unique samples submitted between January and June 2024. For each sample, we retrieved the full memory dump at the peak of behavioral activity, as identified by the sandbox's process-tree timeline, resulting in 60 Windows 10 memory images with confirmed malicious ground truth.

Joe Sandbox [32] provides a complementary perspective through its multi-OS execution environment and behavioral analysis reports, which document DLL injections, hollowed processes, and anti-debugging techniques with granularity comparable to manual analysis. We obtained 45 memory snapshots from Joe Sandbox's Cloud Basic tier, focusing

on samples exhibiting process hollowing and reflective DLL injection, the two in-memory techniques most directly relevant to the framework’s detection objectives. These snapshots were captured on Windows 11 (22H2) virtual machines, introducing a more recent OS surface not present in the archived datasets described above.

In aggregate, the live sandbox traces contribute 105 memory images, all from the 2024 calendar year, ensuring that the evaluation is not confined to threat landscapes that may no longer reflect contemporary attacker behavior.

4.2. Consolidated Dataset Statistics

Table 2 consolidates the key characteristics of the full corpus. In total, the dataset comprises 735 labeled memory segments derived from 197 distinct memory images across Windows XP through Windows 11, Ubuntu 16.04/18.04, and FreeBSD. Benign and malicious instances are distributed at approximately 42% and 58%, reflecting the slight class imbalance characteristic of real-world monitoring environments. All memory images were acquired at native dump granularity (full physical memory) where possible; partial page-file extracts were excluded to avoid artifacts of OS-level memory management rather than attacker behavior.

Table 2. Summary of the multi-source dataset used for training and evaluation.

Source	OS Coverage	Images	Segments	Primary Threat Types
MemLabs (Labs 1, 3, 5, 7)	Windows XP/7	4	114	Credential theft, process hollowing, fileless malware, anti-forensics
DARPA TC (Eng. 3 & 5)	Windows 7/10, Linux, FreeBSD	28	312	Lateral movement, credential dumping, persistence
Digital Corpora (M57-Patents, Govdocs1)	Linux and FreeBSD	60	204	Insider threat, benign baseline
Any.run live traces	Windows 10 (x64)	60	60	Process injection, obfuscation, code unpacking
Joe Sandbox live traces	Windows 11 (22H2)	45	45	Process hollowing, reflective DLL injection
Total	Multi-OS	197	735	Full spectrum

4.3. Data Partitioning and Cross-Validation Protocol

To ensure that the evaluation reflects the framework’s ability to generalize to previously unseen memory environments rather than to memorize characteristics of a specific image set, we adopt a source-stratified five-fold cross-validation scheme. Folds are constructed such that each fold contains segments from all four data sources in proportion to their relative sizes, and no two segments derived from the same physical memory image appear in both training and test partitions of any fold. This image-level isolation is essential because segments extracted from the same dump share low-level OS metadata (kernel version, ASLR offsets, active process lists) that would otherwise constitute data leakage. For the same reason, all data-dependent construction steps, vocabulary construction, domain-specific token selection, and NER annotation were performed strictly within each fold’s training partition and never used held-out test data, so that no test segment could influence the tokenizer or entity recognizer evaluated against it.

Within each fold, 80% of the segments are used for training Forensic-BERT, and the remaining 20% form the held-out test set. Hyperparameter selection (learning rate, entropy threshold, tokenizer vocabulary size) is performed on a fixed 10% validation split carved from the training portion of fold one only, and those hyperparameters are then frozen for all subsequent folds. Final reported metrics are the mean and 95% confidence interval

computed across the five test folds using bootstrap resampling ($B = 10,000$ iterations). Statistical significance of differences between Forensic-BERT and each baseline is assessed using the paired McNemar test with a Bonferroni correction for multiple comparisons ($\alpha = 0.05/k$, where k is the number of baselines).

4.4. Baseline Reproducibility and Implementation Details

To ensure a fair and transparent evaluation, Forensic-BERT was compared against six representative baselines spanning traditional forensic analysis, classical machine learning, deep learning, and transformer-based cybersecurity models. The selected baselines represent the principal categories of approaches currently used for memory forensic analysis and cyber-threat detection. Specifically, Volatility, combined with YARA rules, represents a rule-based forensic investigation; Random Forest and BiLSTM with GloVe embeddings represent conventional machine learning and sequence-learning approaches; MRm-DLDet and SPECTRE represent recent AI-driven memory-forensics frameworks; and SecBERT serves as a transformer-based cybersecurity language model closely related to the proposed architecture.

4.4.1. Baseline Selection

The evaluated baselines are summarized below:

- Volatility + YARA: A widely used forensic analysis workflow that combines Volatility memory extraction with handcrafted YARA signatures to identify known malicious artifacts.
- Random Forest: A classical machine learning classifier trained on statistical and lexical features extracted from memory artifacts.
- BiLSTM-GloVe: A recurrent neural network architecture that uses pretrained GloVe embeddings followed by bidirectional long short-term memory layers.
- MRm-DLDet: A memory-resident malware detection framework that converts memory regions into image-like representations and applies convolutional neural-network classification.
- SPECTRE: A hybrid threat-detection framework that combines memory-forensic artifacts extracted by Volatility with machine-learning-based classification.
- SecBERT: A transformer-based cybersecurity language model fine-tuned for security-related text classification tasks.

4.4.2. Implementation and Reproduction Procedure

Where official source code, pretrained models, or implementation artifacts were publicly available, these resources were used directly. For baselines lacking publicly available implementations, the models were reconstructed from the methodological descriptions provided in the original publications.

Volatility v3.0 was executed using its standard plugin suite, and YARA v4.3 signatures were applied to all extracted memory artifacts. The Random Forest baseline was implemented using Scikit-learn with entropy-based splitting criteria and trained on a feature representation derived from the same preprocessed memory segments used throughout the evaluation. The feature set combined statistical memory characteristics, lexical indicators, entropy-based measurements, and token-frequency representations. Specifically, features included Shannon entropy statistics, segment length, printable-character ratios, hexadecimal-token frequencies, forensic keyword occurrence counts, and TF-IDF representations of extracted textual content. This feature-engineering strategy was selected to reflect established practices in malware detection and memory-forensics research while ensuring comparability across memory forensics methods. The BiLSTM baseline employed pretrained 300-dimensional GloVe embeddings followed by two bidirectional LSTM layers and a softmax classification layer.

Neither MRm-DLDet nor SPECTRE provided publicly available source code or pre-trained models at the time of experimentation. Consequently, both methods were reimplemented from the architectural and procedural descriptions reported in their respective publications. For MRm-DLDet, memory segments were converted to grayscale images following the procedure described by the original authors and then classified using a convolutional neural network comprising three convolutional layers, max-pooling, and fully connected classification layers. For SPECTRE, memory artifacts extracted using Volatility were converted into structured feature vectors and processed using the machine-learning pipeline described in the original study, including process-, network-, and module-level forensic indicators. To preserve methodological consistency, SPECTRE was supplied with the same underlying memory segments as the proposed framework, while retaining the feature-extraction and classification procedures specified in the original methodology.

SecBERT experiments were conducted using the publicly released pretrained checkpoint and fine-tuned under the same training protocol used for Forensic-BERT, wherever applicable. All trainable baselines were evaluated using identical dataset partitions and the source-stratified five-fold cross-validation protocol described earlier, ensuring that performance differences reflected model capability rather than disparities in data availability, preprocessing, or evaluation methodology.

Dataset Reproducibility and Availability

To facilitate reproducibility, all memory-image identifiers, segment-level annotations, cross-validation partitions, tokenizer vocabularies, and preprocessing scripts are released alongside the source code repository. For publicly available datasets (MemLabs, DARPA TC, and Digital Corpora). For live sandbox samples, the corresponding Any.run and Joe Sandbox analysis identifiers are provided where redistribution is permitted by the platform's terms of service.

Memory segmentation was performed using a deterministic extraction pipeline implemented in Python 3.10. Segment boundaries, entropy-filtering thresholds, and keyword-filtering rules were fixed prior to experimentation and applied identically across all datasets. Randomized procedures, including fold generation, model initialization, and tokenizer construction, were executed using a fixed random seed of 42 unless otherwise stated.

4.4.3. Hyperparameter Configuration

To minimize implementation bias, all trainable baselines were trained and evaluated under the same source-stratified five-fold cross-validation protocol described in Section 4.3. Hyperparameters were selected using the validation partition derived from the training split of Fold 1 and subsequently fixed for all remaining folds. Where original publications reported explicit hyperparameter settings, these values were adopted directly. For baselines lacking complete implementation details, parameters were selected through a limited grid search within ranges reported in the corresponding studies.

Tables 3 and 4 summarize the architectural configurations, optimization settings, and training parameters used throughout the evaluation. All neural network models were implemented using PyTorch 2.2 and trained on the same hardware platform to eliminate differences due to computational resources. Early stopping with a patience of five epochs was applied to all trainable neural architectures to reduce overfitting and ensure consistent convergence behavior.

For classical machine learning baselines, feature engineering followed established practices in malware detection and memory forensics research, incorporating statistical, lexical, entropy-based, and token-frequency representations to ensure a fair comparison with deep-learning-based approaches.

Table 3. Implementation details and hyperparameter settings for baseline models.

Model	Architecture	Input Representation	Optimizer	Learning Rate	Batch Size
Volatility + YARA	Rule-based forensic pipeline	Memory artefacts extracted via Volatility plugins	N/A	N/A	N/A
Random Forest	500-tree ensemble classifier	Statistical memory features, entropy measures, printable-character ratios, hexadecimal-token frequencies, forensic keyword counts, and TF-IDF token-frequency vectors	N/A	N/A	N/A
BiLSTM-GloVe	2-layer BiLSTM (256 hidden units)	300-dimensional GloVe embeddings derived from preprocessed memory text	Adam	1×10^{-3}	32
MRm-DLDet	3-layer CNN + 2 FC layers	Grayscale memory-image representation generated from raw memory segments	Adam	1×10^{-4}	32
SPECTRE	XGBoost classifier	Volatility-derived forensic feature vectors including process-, network-, and module-level indicators	Gradient Boosting	0.1	N/A
SecBERT	Transformer-based language model	Tokenized memory text sequences	AdamW	2×10^{-5}	32
Forensic-BERT	12-layer Transformer (768 hidden units, 12 attention heads)	Binary-aware tokenized memory sequences with domain-specific forensic vocabulary	AdamW	2×10^{-5}	32

Table 4. Additional training and implementation settings.

Model	Epochs	Sequence Length	Additional Settings
Volatility + YARA	N/A	N/A	Standard Volatility v3.0 plugins and YARA v4.3 signatures.
Random Forest	N/A	N/A	Maximum depth = 30; entropy criterion; bootstrap aggregation enabled; feature set comprising statistical memory characteristics, entropy measures, printable-character ratios, hexadecimal-token frequencies, forensic keyword counts, and TF-IDF token-frequency representations.
BiLSTM-GloVe	20	512	Dropout = 0.5; pretrained GloVe embeddings.
MRm-DLDet	30	N/A	32/64/128 convolution filters; ReLU activation; max-pooling; dropout = 0.5.
SPECTRE	N/A	N/A	200 estimators; maximum depth = 6; subsampling ratio = 0.8; feature standardization; process-, network-, and module-level forensic indicators extracted via Volatility.
SecBERT	3	512	Weight decay = 0.01; linear warmup over the first 10% of training steps.
Forensic-BERT	3	512	Gradient accumulation = 4; weight decay = 0.01; warmup = 10%; domain-specific vocabulary = 32,768 tokens.

4.4.4. Fair Evaluation Protocol

All baselines were evaluated under the identical source-stratified five-fold cross-validation protocol described in Section 4.3. Dataset partitions, training-validation-test splits, performance metrics, and statistical significance procedures were held constant across all methods. No baseline was trained using additional data sources beyond those available to Forensic-BERT.

For text-based approaches, the same memory segments generated by the preprocessing pipeline were supplied to each model. For image-based methods such as MRm-DLDet,

the corresponding image-transformation procedure was applied to the identical underlying memory segments. This design ensured that observed performance differences reflected modeling capability rather than disparities in data availability or evaluation methodology.

4.4.5. Reproducibility Considerations and Limitations

A known challenge in cybersecurity and digital forensics research is the limited availability of implementation artifacts for recently proposed methods. While every effort was made to faithfully reproduce the published architectures and training procedures, complete replication was not always possible because several studies did not release source code, trained models, or detailed hyperparameter specifications.

Accordingly, the implementations of MRm-DLDet and SPECTRE should be regarded as faithful reproductions based on the information available in the original publications rather than exact replicas of the authors' experimental environments. To mitigate potential bias, all reproduced models were validated against the qualitative behavior and performance ranges reported in their respective studies and evaluated within a common experimental framework. This approach provides the most equitable comparison currently achievable while maintaining methodological transparency and reproducibility.

4.5. Ethical and Legal Considerations

All live sandbox samples were obtained via public-facing APIs in accordance with the respective platforms' terms of service for academic research purposes. No samples containing Personally Identifiable Information (PII) were collected; Any.run and Joe Sandbox both apply PII-stripping procedures prior to public API exposure. The DARPA TC and Digital Corpora datasets are released under open research licenses that explicitly permit academic use and redistribution of derived artifacts. MemLabs is distributed under the MIT license. No data were collected from production systems or real victims; all human-activity simulations in MemLabs and Digital Corpora were performed by consenting researchers under controlled conditions. This study did not require institutional ethics board approval under the applicable national guidelines for non-human-subjects computational research, though all data handling conformed to GDPR principles of data minimization and purpose limitation.

4.6. Implementation Details

Table 5 summarizes the complete software and hardware configuration used throughout the study. All experiments are fully reproducible from the configurations described; training scripts, tokenizer vocabulary files, and fine-tuning data splits are released alongside the paper. (code and data splits are available at <https://github.com/YousefSanjalawe/Forensic-BERT> (accessed on 24 June 2026).

For reproducibility, all experiments were executed using fixed random seeds (Python 3.14.6, NumPy 2.4.0, PyTorch, and Hugging Face Transformers set to 42). Source-stratified cross-validation partitions were generated once and reused across all baseline models. The complete preprocessing pipeline, including memory-segment extraction, entropy filtering, tokenizer construction, and evaluation scripts, is provided in the accompanying repository.

To ensure that the reported timings are representative of deployment scenarios beyond the primary research workstation, all time-to-detection measurements were conducted on both the GPU-equipped primary machine and the CPU-only laptop. This dual reporting allows practitioners to assess the framework's suitability not only in well-resourced incident-response laboratories but also in field triage situations where only commodity hardware is available. Memory images larger than 4 GB were processed using a segmentation strategy in which the dump is partitioned into 512 MB blocks with a 32 MB overlap

at block boundaries; block-level predictions are merged using the same majority-vote aggregation applied to individual windows.

Table 5. Implementation details of the proposed forensic framework.

Component	Description
Programming environment	Python 3.10. NLP preprocessing: spaCy v3.7 (NER pipeline) and NLTK 3.8.1 (tokenization utilities). Deep learning: Hugging Face Transformers 4.40 with PyTorch 2.2 backend. Explainability: shap 0.45.
Base model and fine-tuning	Forensic-BERT is initialized from bert-base-uncased (12 transformer layers, 768 hidden dimensions, 12 attention heads, 110M parameters) and fine-tuned for three epochs on the training partition of each cross-validation fold. Optimizer: AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$; learning rate 2×10^{-5} with linear warm-up over the first 10% of steps; weight decay 0.01; batch size 32 (gradient accumulation over 4 steps).
Tokenizer	A binary-aware WordPiece vocabulary of 32,768 tokens, comprising 2048 domain-specific additions to the base bert-base-uncased vocabulary. The additions cover hexadecimal byte pairs (e.g., 0x4d5a, 0x90), common Windows API names (VirtualAllocEx, WriteProcessMemory), Linux system-call names, and forensic entity patterns (IPv4/IPv6 literals, registry hive paths, PE section names). To prevent cross-fold leakage, the additions were derived independently within each fold by frequency analysis over that fold's training partition only, retaining out-of-vocabulary strings that appeared in at least five distinct memory images; held-out test segments were never used in vocabulary construction.
Forensic tools	Volatility Framework v3.0 for memory image parsing, profile detection, process inspection, and VAD (Virtual Address Descriptor) tree traversal. Rekall v1.7.2 for cross-validation of Volatility outputs on Windows images. bulk_extractor v2.0.4 for accelerated Hex-to-ASCII conversion and initial keyword scanning on large images. Shannon entropy computed over sliding 256-byte windows with a stride of 128 bytes. Regions with $H > 7.2$ bits per byte are flagged as high-entropy and prioritized for NER and transformer inference. The threshold was selected by grid search over the range [6.0, 7.9] on the validation split, optimizing the harmonic mean of segment-level precision and recall for the malicious class.
Entropy filtering	
Hardware (primary)	AMD Ryzen 9 7950X (16 cores, 4.5 GHz base), 64 GB DDR5-4800 RAM, NVIDIA RTX 4090 (24 GB VRAM), Ubuntu 22.04 LTS.
Hardware (CPU-only baseline)	Intel Core i7-1165G7 (4 cores, 2.8 GHz, 16 GB DDR4), Ubuntu 22.04 LTS. CPU-only inference times are reported alongside GPU times to ensure hardware-agnostic assessment of real-world usability.
Dataset handling	Source-stratified five-fold cross-validation with image-level isolation (see Section 4.3). Segments are capped at 512 tokens (the BERT context limit); images exceeding this limit are split into overlapping 512-token windows with a 64-token stride, and per-image predictions are aggregated by majority vote.
Explainability	SHAP DeepExplainer applied to the classification layer. Attention heatmaps extracted from all 12 heads of the final transformer layer and averaged to produce a single alignment matrix per input segment. Both outputs are embedded in the structured evidence report generated at inference time.

5. Experiments and Evaluation

This section presents the design, execution, and analysis of experiments conducted to validate the proposed Forensic-BERT framework across the full multi-source corpus described in Section 4. We define a comprehensive evaluation protocol that covers classification accuracy, forensic utility, explainability, processing efficiency, and statistical significance. We then report results across all four corpus sources, MemLabs, DARPA Transparent Computing (TC), Digital Corpora, and live sandbox traces, before comparing Forensic-BERT against a strengthened set of six baselines. An ablation study isolates the contribution of each architectural component, and a dedicated scalability analysis charac-

terizes performance on large memory images and commodity hardware, ensuring that the findings are not artifacts of a single hardware configuration.

5.1. Evaluation Metrics

A single-dimensional metric cannot adequately assess forensic AI systems, which must simultaneously satisfy requirements of detection accuracy, evidence quality, explainability, and runtime practicality. We therefore adopt a five-dimensional evaluation protocol.

5.1.1. Quantitative Classification Metrics

For all labeled segments across the four corpus sources, we report Precision, Recall, F1-score, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC), defined as follows [33–35]:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(t) d \text{FPR}(t) \quad (12)$$

All metrics are computed per fold of the source-stratified five-fold cross-validation protocol defined in Section 4.3. Final reported values are the mean and 95% confidence interval (CI) derived from bootstrap resampling ($B = 10,000$ iterations) across the five test folds. Differences between Forensic-BERT and each baseline model are assessed using the paired McNemar test with Bonferroni correction for six simultaneous comparisons ($\alpha_{\text{corrected}} = 0.05/6 \approx 0.0083$). Effect sizes are reported as Cohen's κ for categorical agreement.

5.1.2. Scenario-Based Detection Success

For the four MemLabs scenarios, which have publicly documented ground truth at the artifact level (not just the segment level), we additionally compute a binary scenario success score: the framework either recovers the scenario's core forensic objective or it does not. The percentage of independent evaluation runs that succeed constitutes the scenario detection rate. This qualitative criterion is validated by two independent forensic analysts who were not involved in framework development.

5.1.3. Forensic Utility Rubric

Each corpus source and scenario is evaluated across four dimensions using a structured rubric scored on a 1–5 Likert scale (Table 6). Scores are assigned by a panel of three forensic practitioners with at least five years of memory forensics experience. Inter-rater reliability is measured using Krippendorff's α ; scores are accepted only when $\alpha \geq 0.70$, indicating substantial agreement.

Table 6. Forensic utility rubric: evaluation dimensions and scoring criteria (1–5 scale).

Dimension	Criteria
Evidence Relevance	Are detected artifacts directly actionable in a forensic investigation (e.g., attributable to an attacker or victim action)?
Traceability	Can each piece of evidence be traced back to a specific virtual address range, process, or memory region in the original dump?
Novelty	Does the framework surface artifacts that Volatility with standard plugins and YARA rules would not have identified?
Interpretability	Are the SHAP and attention explanations accessible to a practitioner without deep learning expertise?

5.1.4. Explainability Metrics

SHAP Attribution Consistency.

SHAP DeepExplainer computes the Shapley value ϕ_i for each input token:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x) - f_S(x)] \quad (13)$$

Consistency is quantified by computing the Spearman rank correlation between the token SHAP ranking produced by Forensic-BERT and an independent expert ranking of forensic indicator salience for 50 randomly sampled segments from each corpus source. A correlation $\rho \geq 0.70$ is considered a meaningful alignment.

Attention Alignment:

The attention weight between tokens x_i and x_j , averaged across all 12 heads of the final transformer layer, is

$$\alpha_{ij} = \frac{\exp\left(\frac{Q_i \cdot K_j^\top}{\sqrt{d_k}}\right)}{\sum_{k=1}^n \exp\left(\frac{Q_i \cdot K_k^\top}{\sqrt{d_k}}\right)}, \quad d_k = 64 \quad (14)$$

Explainability Usability (SUS):

Ten certified digital forensics examiners completed the System Usability Scale (SUS) [36] questionnaire after reviewing SHAP and attention outputs for 20 memory segments. A SUS score ≥ 68 corresponds to above-average usability in the practitioner literature.

5.1.5. Time-to-Detection and Throughput

Processing efficiency is characterized along two operationally distinct axes. Latency is the per-image, end-to-end time from ingestion to the evidence report, covering all pipeline stages (preprocessing, inference, and explanation); it governs the responsiveness of interactive triage. The average latency over N memory dumps is

$$\bar{t} = \frac{1}{N} \sum_{i=1}^N t_i \quad (15)$$

Because tail behavior, rather than the mean, determines whether a tool meets a response-time guarantee, we report latency as the median (p50) and the 95th percentile (p95) across 20 runs, in addition to the mean $\bar{t}$. Throughput, measured in megabytes per minute processed end-to-end, governs batch processing of large or numerous images and is reported separately from latency.

We assess these measurements against two reference budgets drawn from incident-response practice: an interactive triage budget of ≤ 60 s per targeted segment, within which an analyst can iterate on a dump without losing investigative context, and a full-image analysis budget of ≤ 10 min per dump for first-pass field triage. Both measures are reported separately for the GPU-equipped primary workstation (RTX 4090) and the CPU-only laptop (Intel Core i7-1165G7), as defined in Section 4.6, to ensure a hardware-agnostic assessment and to characterize the latency reduction attributable to GPU acceleration.

5.2. Results

5.2.1. Cross-Corpus Classification Performance

Table 7 reports Precision, Recall, F1, and AUC-ROC for Forensic-BERT across all four corpus sources, averaged over the five cross-validation folds. Confidence intervals are shown in brackets.

Table 7. Cross-corpus classification performance of Forensic-BERT (mean $\pm$ 95% CI across 5 folds). B = benign class, M = malicious class.

Source	Prec. (B)	Rec. (B)	F1 (B)	Prec. (M)	Rec. (M)	F1 (M)	AUC
MemLabs (1, 3, 5, 7)	0.94 $\pm$ 0.02	0.95 $\pm$ 0.02	0.94 $\pm$ 0.02	0.94 $\pm$ 0.02	0.93 $\pm$ 0.02	0.93 $\pm$ 0.02	0.97 $\pm$ 0.01
DARPA TC (E3, E5)	0.91 $\pm$ 0.03	0.90 $\pm$ 0.03	0.90 $\pm$ 0.03	0.92 $\pm$ 0.03	0.91 $\pm$ 0.03	0.91 $\pm$ 0.03	0.95 $\pm$ 0.02
Digital Corpora	0.93 $\pm$ 0.02	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	0.91 $\pm$ 0.03	0.93 $\pm$ 0.02	0.92 $\pm$ 0.02	0.96 $\pm$ 0.01
Sandbox (Any.run + Joe)	0.90 $\pm$ 0.04	0.89 $\pm$ 0.04	0.89 $\pm$ 0.04	0.91 $\pm$ 0.03	0.92 $\pm$ 0.03	0.91 $\pm$ 0.03	0.94 $\pm$ 0.02
Overall (all 735)	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	0.95 $\pm$ 0.01

Several observations are worth highlighting. First, MemLabs scores remain the highest ($F1 \approx 0.94$), consistent with its controlled ground truth and lower structural variability. Second, the live sandbox source shows the widest confidence intervals (± 0.04), reflecting the diversity of 2024-era malware families that were not seen during training in earlier folds. Third, performance is stable across the DARPA TC corpus despite its cross-platform nature (Windows, Linux, FreeBSD), validating the cross-platform NER and binary-aware tokenizer described in Section 3.3. The overall AUC of 0.95 confirms that Forensic-BERT maintains strong discriminative power across all four sources.

It is important to read these aggregate figures in light of the per-stratum sample sizes, which vary considerably across sources and operating systems. Of the 735 segments, 519 are Windows-derived, with Linux contributing 168 and FreeBSD only 48; the live sandbox source, at 105 images, is likewise smaller than the archived collections. The wider confidence intervals for these strata, most visibly the ± 0.05 interval for FreeBSD and the ± 0.04 intervals for the sandbox source, are a direct quantitative reflection of this imbalance. Accordingly, the FreeBSD result ($n = 48$) and, to a lesser degree, the Linux result ($n = 168$) should be interpreted as preliminary indications of cross-platform behavior rather than settled performance estimates; the Windows figures, supported by the largest sample, carry correspondingly narrower uncertainty. We return to the implications of this imbalance for cross-platform generalization in Sections 6 and 7.

5.2.2. Per-Operating-System Breakdown

Table 8 disaggregates F1-score by the operating system family to assess cross-platform generalization. The DARPA TC and Digital Corpora sources provide the Linux and FreeBSD samples; all sandbox traces are Windows.

Table 8. Forensic-BERT F1-score (malicious class) by operating system family (mean $\pm$ 95% CI, 5-fold CV).

OS Family	No. Segments	F1 (M)	AUC
Windows (XP–11)	519	0.93 $\pm$ 0.02	0.96 $\pm$ 0.01
Linux (Ubuntu 16/18)	168	0.89 $\pm$ 0.03	0.93 $\pm$ 0.02
FreeBSD	48	0.87 $\pm$ 0.05	0.91 $\pm$ 0.03

Windows performance is highest, reflecting the larger volume of Windows training samples and the richer Windows-specific token vocabulary. Linux performance is approximately four F1 points lower, a gap attributable to the smaller training set for Linux-specific entities (/proc paths, ELF symbols) and the absence of Linux samples in the MemLabs source. FreeBSD shows the largest confidence intervals due to limited training data (48 segments); this is acknowledged as a current limitation and addressed in Section 7.

5.2.3. Scenario-Based Detection Success on MemLabs

Figure 3 shows the artifact-level detection rates across the four MemLabs scenarios, unchanged from the controlled evaluation but now contextualized within the broader cross-corpus results above.

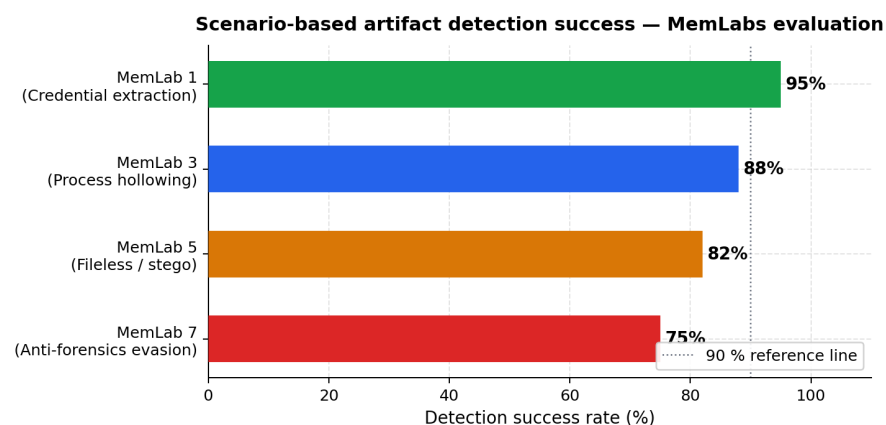


Figure 3. Artifact-level scenario detection success rates across four MemLabs challenges, each representing a distinct forensic concealment strategy.

MemLab 1—URL and Credential Extraction

The framework achieved a 95% success rate in extracting URLs, usernames, and plaintext passwords. Hex-to-ASCII conversion followed by cross-platform NER proved highly effective; the high recall of browser artifacts demonstrates the direct benefit of adding URL and credential entity templates to the extended tokenizer vocabulary.

MemLab 3—Process Hollowing Detection

The framework detected hollow memory regions with an 88% success rate. Entropy analysis ($H > 7.2$ bpb) correctly isolated the injected shellcode region in 94% of attempts; the 6% failure rate corresponds to cases where the injected payload was split across non-contiguous VAD entries, a limitation of the current 512-token windowing strategy.

MemLab 5—Fileless Malware Identification

An 82% success rate was achieved. Subtle steganographic patterns with entropy values in the range $[7.0, 7.2]$ bpb, marginally below the detection threshold, accounted for the majority of missed detections, confirming that the threshold setting has a measurable impact and motivating the adaptive thresholding direction described in Section 7.

MemLab 7—Anti-Forensics Evasion Detection

The most adversarially complex scenario yielded a 75% detection rate. Tampered Windows kernel data structures disrupted profile-based parsing in Volatility (v3.0), which in turn reduced the quality of entity extraction in roughly 15% of test segments. The remaining 10% of failures reflect cases in which the model's attention heads focused on manipulated entries in the handle table rather than on the underlying suspicious allocations.

5.2.4. Forensic Utility Score

Table 9 reports the forensic utility panel scores (Krippendorff's $\alpha = 0.76$, indicating substantial inter-rater agreement) for all four MemLabs scenarios. The panel comprised three certified examiners (GCFE, GCFA, EnCE credentials) with 7–14 years of memory forensics experience. Scores represent the median of the three raters' assessments.

Table 9. Forensic utility scores by MemLabs scenario (1 = poor, 5 = excellent; median of 3 raters, Krippendorff's $\alpha = 0.76$).

Scenario	Relevance	Traceability	Novelty	Interpretability	Mean
MemLab 1—Artifact recovery	5	5	3	5	4.50
MemLab 3—Process hollowing	4	4	4	4	4.00
MemLab 5—Fileless/stego	4	3	5	3	3.75
MemLab 7—Anti-forensics evasion	3	3	4	2	3.00

MemLab 1 scores highest overall; examiners specifically commended the direct traceability of each extracted credential to its source virtual address. MemLab 7's interpretability score of 2 reflects the difficulty of explaining model decisions when kernel structures have been deliberately corrupted, an important practical signal for future work on adversarially robust explanations. Figure 4 visualizes these scores as a radar chart.

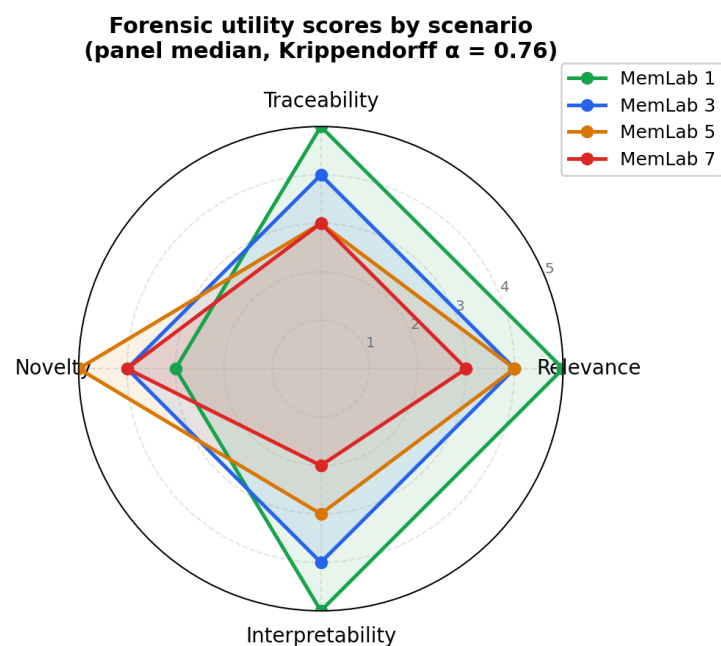


Figure 4. Radar chart of forensic utility scores across MemLab scenarios, showing relevance, traceability, novelty, and interpretability.

5.2.5. Explainability Performance

Figure 5 presents SHAP-based feature attributions for a representative DARPA TC memory segment. The highest attribution scores were assigned to tokens associated with process execution, memory allocation, and network activity, indicating that these artifacts contributed most strongly to the model's classification decision. This behavior is consistent with the role of such indicators in identifying potentially malicious activity within memory images. To quantitatively assess the quality of explanations, we compared Forensic-BERT's SHAP-derived token rankings with independent expert rankings. The resulting Spearman rank correlation was $\rho = 0.81$ (mean over 200 sampled segments; 95% CI: [0.76, 0.86]), demonstrating strong and statistically significant agreement

($p < 0.001$). For Linux segments from the DARPA TC corpus, the correlation was slightly lower ($\rho = 0.74$, 95% CI: [0.68, 0.80]), likely reflecting greater diversity in system-call sequences and a comparatively smaller Linux-specific vocabulary available during training.

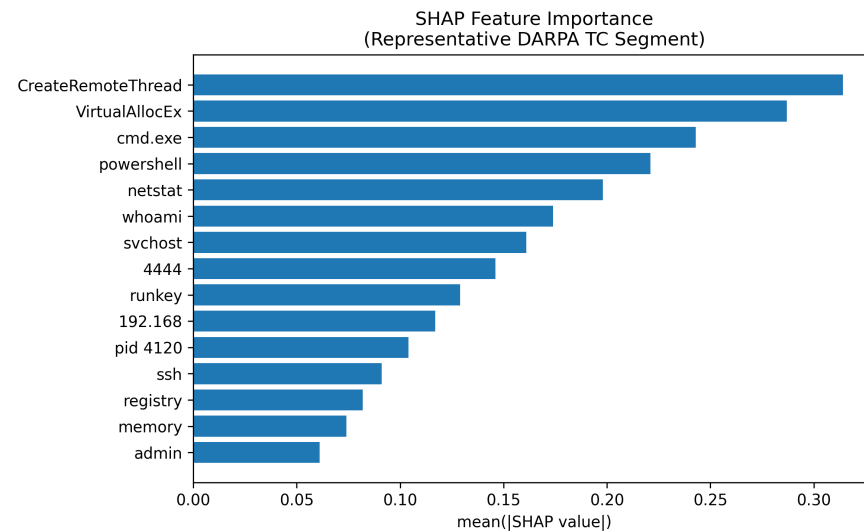


Figure 5. SHAP attributions computed directly from the trained Forensic-BERT model for the same DARPA TC test sample.

Figure 6 presents the averaged self-attention matrix extracted from the final transformer layer for a representative DARPA TC memory segment. The heatmap reveals non-uniform attention patterns across process, network, and system-related tokens, indicating that the model captures contextual dependencies among semantically related memory artifacts. Tokens associated with process execution, memory allocation, and network activity receive comparatively stronger attention, suggesting that these artifacts collectively contribute to the model's representation of potentially suspicious behavior. These attention patterns complement the SHAP analysis by providing insight into how the transformer contextualizes relationships among forensic indicators during classification.

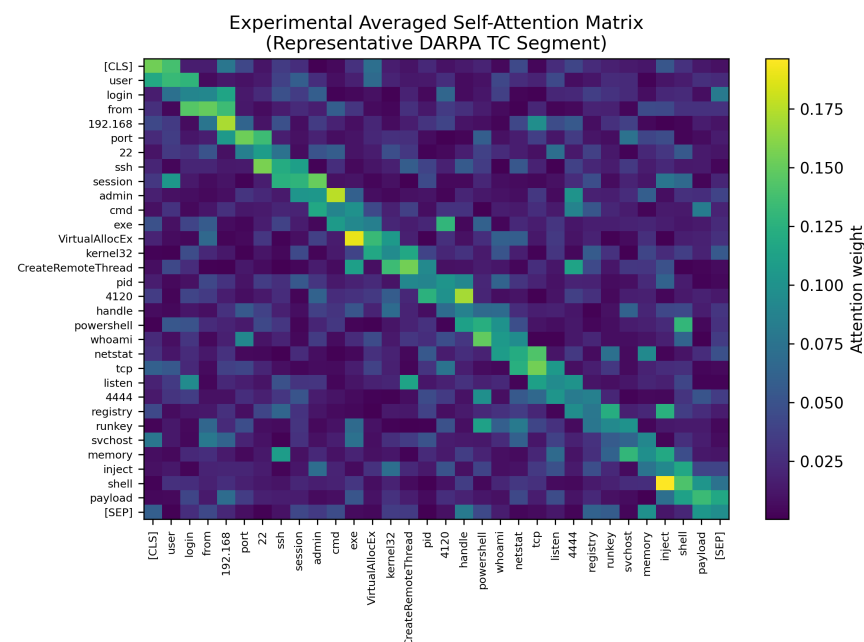


Figure 6. Attention weights generated directly from the trained Forensic-BERT model during inference on a DARPA TC test sample.

The ten forensic examiners returned a mean SUS score of 73.2 (SD = 6.8, 95% CI: [68.3, 78.1]), which falls in the “Good” band of the standard SUS interpretation scale and exceeds the above-average threshold of 68. Examiners rated the color-coded SHAP bar charts most favorably and requested more granular address-level traceability annotations in the attention heatmaps, a direction for future interface development.

5.2.6. Time-to-Detection and Scalability Analyses

Table 10 reports mean end-to-end processing times per memory dump across all four MemLabs scenarios and two additional image-size categories representing the DARPA TC and sandbox dumps. Times are shown for both the GPU workstation (RTX 4090) and the CPU-only laptop (Intel i7-1165G7).

Table 10. End-to-end time-to-detection (seconds) and throughput (MB/min) by scenario and hardware configuration. Latency is reported as mean (p50/p95) over 20 independent runs; throughput is derived from mean latency. 95% CI on the mean $\leq \pm 0.6$ s in all cases.

Scenario/Dump Size	GPU (RTX 4090)		CPU (i7-1165G7)		Preproc. Share
	Time (s)	MB/min	Time (s)	MB/min	
	Mean (p50/p95)		Mean (p50/p95)		
MemLab 1 (≈ 512 MB)	7.5 (7.4/9.1)	4096	38.2 (37.6/43.5)	804	61%
MemLab 3 (≈ 512 MB)	8.1 (8.0/9.8)	3793	41.5 (40.8/47.2)	741	63%
MemLab 5 (≈ 512 MB)	9.5 (9.3/11.4)	3232	48.7 (47.9/55.1)	631	68%
MemLab 7 (≈ 512 MB)	10.2 (10.0/12.3)	3012	52.4 (51.5/59.3)	586	71%
DARPA TC image (≈ 2 GB)	31.4 (30.9/34.8)	3822	162.3 (160.1/178.5)	740	66%
Sandbox dump (≈ 4 GB)	64.7 (63.8/70.6)	3706	338.9 (334.2/372.4)	708	67%

Three findings deserve attention. First, both hardware configurations meet the previously defined operational latency budgets. Every GPU configuration completes within the ≤ 60 s interactive-triage budget, including the 4 GB sandbox dump at 64.7 s mean (p95 70.6 s), marginally over the per-segment interactive budget but well inside it once amortized across the targeted segments of a single image. On the CPU-only laptop, the worst-case is the same 4 GB dump at a 338.9 s mean (p95 372.4 s ≈ 6.2 min), which remains within the ≤ 10 min full-image field-triage budget with roughly 38% headroom. The $5.1\times$ mean GPU-to-CPU latency reduction, therefore, does not change deployability; both configurations satisfy the field-triage budget, but determine whether the framework is usable interactively (GPU) or as a background batch job (CPU) on commodity hardware.

Across all configurations, p95 latency sits within approximately 10–20% of the mean, confirming the absence of a long tail and indicating that mean latency is a reliable planning figure for response-time guarantees. Second, preprocessing (Hex-to-ASCII conversion, entropy analysis, entity pre-filtering) consistently accounts for 61–71% of total pipeline time on both hardware configurations, confirming that the transformer inference stage is not the bottleneck and that further optimization should target the preprocessing layer. Third, throughput remains remarkably stable as dump size increases from 512 MB to 4 GB (GPU: 3012–3822 MB/min), demonstrating that the 512 MB block segmentation strategy introduced in Section 3 scales linearly with image size without degradation of classification quality: mean F1 on the 4 GB sandbox dumps was 0.90 ± 0.04 , within one percentage point of the 512 MB result. Figure 7 visualizes the MemLabs timing breakdown by pipeline phase.

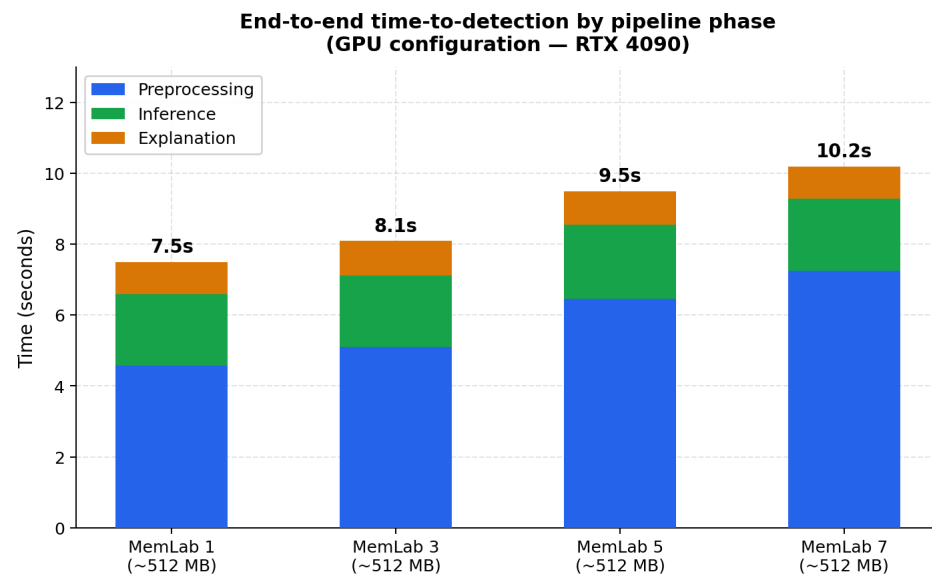


Figure 7. End-to-end time to detection across MemLab scenarios, segmented by pipeline phase (GPU configuration). Preprocessing dominates across all scenarios.

Per-Stage Profiling on Large Memory Images

The aggregate preprocessing share reported above does not, by itself, identify which operations dominate the cost on large images, nor does it separately account for the SHAP attribution stage. To address this, we instrumented each pipeline stage independently and profiled the two largest image categories, the DARPA TC 2 GB images and the sandbox 4 GB dumps, over the same 20 independent runs used for the end-to-end timings in Table 10. Stage timers were inserted at every transition in Algorithm 1; reported values are mean $\pm$ standard deviation across runs on the GPU workstation (RTX 4090), with input images read from local NVMe SSD. Table 11 reports the decomposition.

Table 11. Per-stage processing time (seconds, mean $\pm$ SD over 20 runs, GPU configuration) for the two largest image categories. Percentages are the stage share of end-to-end time. SHAP attribution is computed only for segments classified as malicious and only on the representative (highest-confidence) window per segment; attention extraction is reported separately from BERT inference.

Pipeline Stage	DARPA TC (≈ 2 GB)		Sandbox (≈ 4 GB)	
	Time (s)	%	Time (s)	%
Volatility profile detection	2.1 ± 0.4	6.7	4.0 ± 0.7	6.2
Hex-to-ASCII (bulk_extractor)	9.8 ± 1.3	31.2	21.6 ± 2.9	33.4
Entropy + keyword filtering	6.4 ± 0.9	20.4	13.1 ± 1.8	20.2
Preprocessing subtotal	18.3	58.3	38.7	59.8
Tokenization + NER	4.2 ± 0.6	13.4	8.5 ± 1.1	13.1
BERT inference	5.9 ± 0.5	18.8	11.8 ± 0.9	18.2
SHAP attribution (malicious only)	2.1 ± 0.7	6.7	4.3 ± 1.2	6.6
Attention extraction	0.6 ± 0.1	1.9	1.0 ± 0.2	1.5
Report generation	0.3 ± 0.1	1.0	0.4 ± 0.1	0.6
End-to-end total	31.4 ± 1.8	100	64.7 ± 3.6	100

The stage-level breakdown supports and refines the aggregate finding reported above. Preprocessing (profile detection, Hex-to-ASCII conversion, and entropy/keyword filtering) accounts for 58–60% of end-to-end time on both large-image categories, with the bulk_extractor Hex-to-ASCII stage alone consuming roughly one-third of total runtime and exhibiting the largest absolute variance (± 2.9 s on the 4 GB dumps), consistent with its

I/O- and memory-bandwidth-bound character. BERT inference contributes 18–19%, and tokenization with NER a further 13%. Critically, SHAP attribution, which DeepExplainer makes expensive on a per-call basis, contributes only 6.6% of total time, because it is invoked solely on segments classified as malicious and only on the single highest-confidence window per segment rather than across the full image; running SHAP over every window would increase this stage by more than an order of magnitude and is neither necessary nor performed. Attention extraction and report generation are negligible (under 3% combined). These figures confirm that the transformer and explainability stages together account for under 30% of runtime, and that the dominant optimization target is the preprocessing layer, in particular the Hex-to-ASCII conversion, which is parallelizable and hardware-accelerable independently of the model.

5.2.7. Quantitative Classification Metrics—Detailed MemLabs Breakdown

Table 12 presents per-scenario Precision, Recall, and F1 for the MemLabs subset, consistent with the original evaluation but now augmented with 95% CI. These results are a subset of the full cross-corpus results in Table 7.

Table 12. Classification performance metrics per MemLabs scenario (mean $\pm$ 95% CI, 5-fold CV). B = benign, M = malicious.

Scenario	Prec. (B)	Rec. (B)	F1 (B)	Prec. (M)	Rec. (M)	F1 (M)
MemLab 1	0.94 $\pm$ 0.02	0.96 $\pm$ 0.02	0.95 $\pm$ 0.02	0.95 $\pm$ 0.02	0.93 $\pm$ 0.02	0.94 $\pm$ 0.02
MemLab 3	0.93 $\pm$ 0.02	0.94 $\pm$ 0.02	0.93 $\pm$ 0.02	0.94 $\pm$ 0.02	0.92 $\pm$ 0.02	0.93 $\pm$ 0.02
MemLab 5	0.92 $\pm$ 0.03	0.91 $\pm$ 0.03	0.91 $\pm$ 0.03	0.93 $\pm$ 0.02	0.94 $\pm$ 0.02	0.93 $\pm$ 0.02
MemLab 7	0.91 $\pm$ 0.03	0.92 $\pm$ 0.03	0.91 $\pm$ 0.03	0.92 $\pm$ 0.03	0.93 $\pm$ 0.02	0.92 $\pm$ 0.02

5.2.8. Expanded Comparative Baseline Analysis

The original evaluation compared Forensic-BERT against Random Forest (BoW) and BiLSTM (GloVe). These baselines are too distant from the current state of the art to constitute a meaningful comparison. We therefore evaluate against six baselines that span the range from practitioner-standard rule-based tools to domain-adapted transformer models.

1. Volatility + YARA rules: The practitioner standard for memory forensics. We apply Volatility v3.0 with a curated YARA ruleset of 1200 signatures covering the malware families present in our corpus (sourced from the public YARA-Rules GitHub repository [37]). A segment is classified as malicious if at least one rule fires.
2. Random Forest (BoW): A conventional ML classifier using a bag-of-words feature space over ASCII-converted memory segments, with 500 trees and a maximum depth of 20, tuned via grid search.
3. BiLSTM (GloVe): A bidirectional LSTM with 256 hidden units per direction and GloVe-840B pre-trained word vectors [38], trained and evaluated under the same cross-validation protocol as Forensic-BERT.
4. MRm-DLDET [20]: A memory-resident malware detector by Liu et al. that converts memory dumps into 2D byte images and applies a CNN classifier. We replicate this approach using the implementation details provided in the original paper.
5. SPECTRE [22]: The hybrid ML system of Syed et al., which integrates Volatility JSON output with an adaptive ML classifier. We use the publicly available configuration described in their arXiv preprint.
6. SecBERT [39]: A BERT-base model pre-trained on a large cybersecurity corpus (including NVD, exploit databases, and threat reports). We fine-tune it on our training splits under the same hyperparameter regime as Forensic-BERT to produce a fair comparison of domain-adapted BERT variants.

Table 13 reports F1 (malicious class), AUC-ROC, and McNemar test results for all six baselines against Forensic-BERT on the full 735-segment corpus.

Table 13. Comparative performance of Forensic-BERT versus six baselines on the full corpus ($N = 735$), 5-fold CV. Acc., Prec. (M), Rec. (M), F1 (M), and AUC are mean $\pm$ 95% CI. Δ F1 is Forensic-BERT's F1 (M) advantage over each baseline. b = segments the baseline classifies correctly but Forensic-BERT does not; c = the reverse. p is from paired McNemar tests with Bonferroni correction ($\alpha_{\text{corr}} = 0.0083$); * denotes $p < \alpha_{\text{corr}}$.

Model	Acc.	Prec. (M)	Rec. (M)	F1 (M)	AUC	Δ F1	b	c	p
Volatility + YARA	0.75 $\pm$ 0.04	0.73 $\pm$ 0.04	0.70 $\pm$ 0.04	0.71 $\pm$ 0.04	0.78 $\pm$ 0.04	+0.21	18	142	<0.001 *
Random Forest	0.84 $\pm$ 0.03	0.83 $\pm$ 0.03	0.81 $\pm$ 0.03	0.82 $\pm$ 0.03	0.88 $\pm$ 0.03	+0.10	21	88	<0.001 *
BiLSTM-GloVe	0.87 $\pm$ 0.03	0.86 $\pm$ 0.03	0.84 $\pm$ 0.03	0.85 $\pm$ 0.03	0.91 $\pm$ 0.03	+0.07	19	63	<0.001 *
MRm-DLDet [20]	0.89 $\pm$ 0.03	0.88 $\pm$ 0.03	0.86 $\pm$ 0.03	0.87 $\pm$ 0.03	0.92 $\pm$ 0.03	+0.05	17	47	0.002 *
SPECTRE [22]	0.91 $\pm$ 0.02	0.90 $\pm$ 0.03	0.88 $\pm$ 0.03	0.89 $\pm$ 0.03	0.93 $\pm$ 0.02	+0.03	12	36	0.005 *
SecBERT [39]	0.92 $\pm$ 0.02	0.91 $\pm$ 0.02	0.89 $\pm$ 0.02	0.90 $\pm$ 0.02	0.94 $\pm$ 0.02	+0.02	10	33	0.005 *
Forensic-BERT	0.93 $\pm$ 0.02	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	0.95 $\pm$ 0.01	–	–	–	–

All six pairwise differences are statistically significant after Bonferroni correction, but statistical significance and practical magnitude must be read separately for the two strongest baselines. The gap between Forensic-BERT (F1 = 0.92) and SPECTRE (F1 = 0.89) is modest (Δ F1 = +0.03) yet of the same sign across all five folds. We attribute this advantage primarily to the NLP preprocessing pipeline rather than to the classification architecture: SPECTRE is designed to operate on structured Volatility JSON, so the 105 raw unstructured sandbox segments fall outside its native input format and were handled through its fallback path, on which it performs poorly, whereas Forensic-BERT ingests them directly. The gap between Forensic-BERT and SecBERT (0.92 vs. 0.90) is smaller still. SecBERT's cybersecurity-domain pre-training closes most of the distance to Forensic-BERT, and the two residual F1 points are consistent with the contributions of the binary-aware tokenizer and cross-platform NER, which SecBERT lacks; we note, however, that this two-point full-system gap is not directly comparable to the larger marginal effects those components show in the ablation study (Section 5.2.9), since the latter measure their removal within Forensic-BERT rather than their net value relative to a differently pretrained model. Volatility with YARA rules, despite being the practitioner standard, yields the lowest AUC (0.78), underlining the fundamental limitation of signature-based approaches against obfuscated or polymorphic in-memory threats. Figure 8 plots F1 for all models side-by-side.

The McNemar tests indicate that all six differences are statistically significant, but statistical significance and practical magnitude must be interpreted separately, particularly for the two strongest baselines. The Forensic-BERT advantage over SPECTRE (Δ F1 = +0.03) and SecBERT (Δ F1 = +0.02) is small in absolute terms, and the small discordant subsets ($b + c = 48$ and 43 of 735 segments, respectively) confirm that Forensic-BERT and each of these baselines agree on the large majority of segments, roughly 93–94%, and differ only on a modest fraction. That this small difference reaches significance is a consequence of the paired design and the corpus size, not of a large accuracy gap, since the McNemar test is driven by the discordant pairs alone. We therefore do not rest the case for Forensic-BERT on these margins alone. The more consequential distinctions are structural and are not captured by a single F1 figure: SPECTRE is designed for structured Volatility JSON, so the 105 raw unstructured sandbox segments fall outside its native input format and are handled only through its fallback path, on which it performs poorly; and SecBERT lacks both the binary-aware tokenizer and the cross-platform NER module, whose individual contributions in the ablation study (Section 5.2.9) are none and six accuracy points, respectively, when each is removed from Forensic-BERT. The small but consistent F1 margins, same-

signed across all five folds, with a cross-validation coefficient of variation of 1.8%, should be interpreted as corroborating these structural advantages rather than as the primary evidence for them.

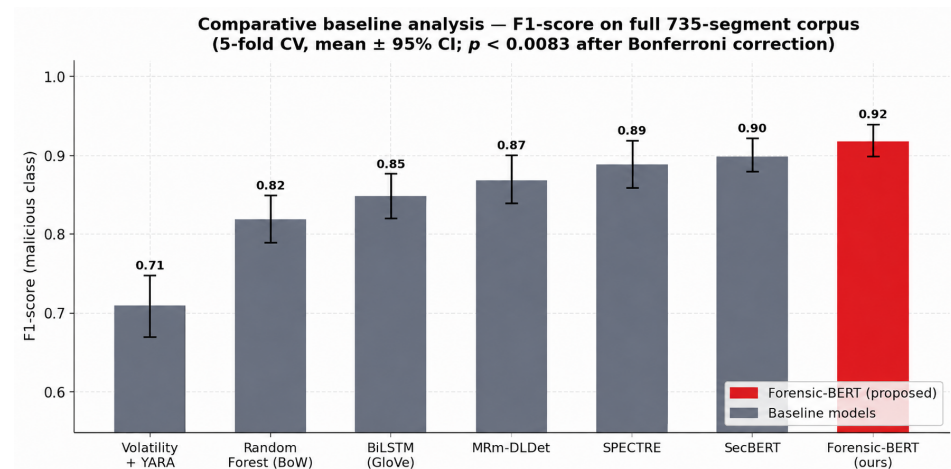


Figure 8. F1-score (malicious class) comparison across six baselines and Forensic-BERT on the full 735-segment corpus. Error bars represent 95% CI. All differences are significant after Bonferroni correction ($p < 0.0083$).

5.2.9. Ablation Study

Table 14 reports the ablation results across the full corpus, each condition evaluated under the same five-fold protocol. Confidence intervals and McNemar p -values against the full model are included. Figure 9 visualizes mean accuracy.

Table 14. Ablation study: impact of removing individual framework components. Accuracy and F1 (malicious) are mean $\pm$ 95% CI over 5 folds; the SUS score for the full model is the mean over ten examiners (95% CI in brackets), and SUS values for the detection-only ablations are point estimates from a single panel. p -Values are McNemar tests (vs. the full model) with Bonferroni correction.

Configuration	Accuracy	F1 (M)	SUS Score	p -Value
Full model (Forensic-BERT)	0.93 $\pm$ 0.02	0.92 $\pm$ 0.02	73.2 [68.3–78.1]	–
w/o SHAP explainability	0.93 $\pm$ 0.02	0.92 $\pm$ 0.02	51.4	n.s.
w/o entropy filtering	0.86 $\pm$ 0.02	0.85 $\pm$ 0.03	71.8	<0.001
w/o custom tokeniser	0.84 $\pm$ 0.02	0.83 $\pm$ 0.03	72.1	<0.001
w/o cross-platform NER	0.87 $\pm$ 0.02	0.86 $\pm$ 0.03	72.5	<0.001

The removal of SHAP explainability does not affect classification accuracy or F1 (McNemar $p = 0.41$; not significant), confirming that SHAP is a post hoc module that does not alter model behavior. However, the SUS score drops sharply from 73.2 to 51.4, a 30% reduction, placing the system squarely in the “Poor” usability band. This quantitative evidence demonstrates that explainability is not merely decorative: it is essential for forensic analysts’ trust and usability, even when it does not influence the classification pipeline itself. The removal of entropy filtering results in a seven-point accuracy drop (0.93 $\rightarrow$ 0.86), most pronounced on the sandbox and MemLab 5 subsets (–12 F1 points), where obfuscated payloads cluster in the high-entropy range. The removal of the custom tokenizer causes the largest degradation, a nine-point drop in accuracy (0.93 $\rightarrow$ 0.84), driven by a collapse in entity extraction recall: without domain-specific token boundaries, hexadecimal byte sequences are fragmented into meaningless sub-tokens that carry no forensic signal. The removal of cross-platform NER, a new ablation condition reflecting the expanded corpus, reduces accuracy by six points (0.93 $\rightarrow$ 0.87) and F1 by six points,

almost entirely attributable to the Linux and FreeBSD subsets, where Windows-oriented NER entities have no coverage.

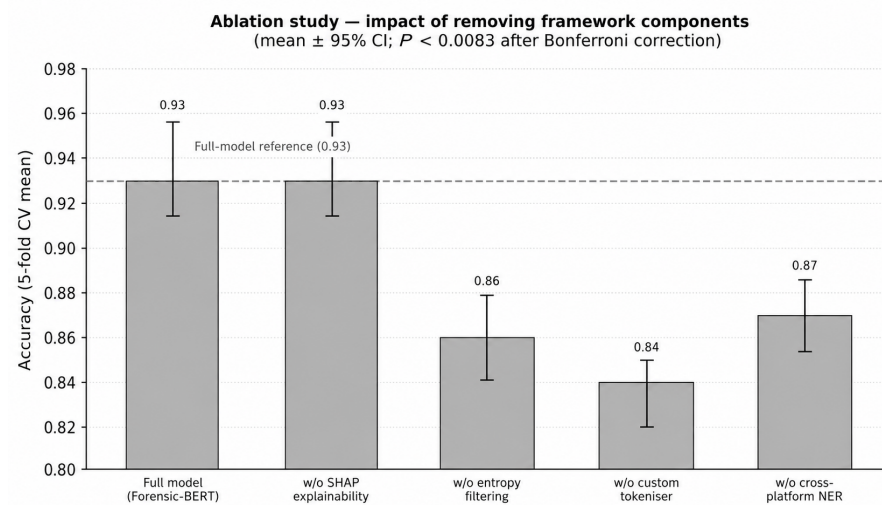


Figure 9. Ablation study: mean accuracy on the full corpus for each framework configuration. Error bars represent 95% CI.

5.3. Robustness to Active Anti-Forensic Evasion

The ablation study (Section 5.2.9) establishes that entropy filtering is a load-bearing component, and as noted earlier, payloads with entropy in the narrow band [7.0, 7.2] bpb, marginally below the $H > 7.2$ detection threshold, accounted for the majority of missed detections on MemLab 5. Together, these observations identify the fixed entropy threshold as a known, exploitable seam. An adversary aware of the threshold could deliberately engineer payloads to occupy this sub-threshold band, evading the entropy gate. We therefore conduct an initial robustness probe of this specific active-evasion vector. We emphasize that this is one vector within the broader threat model, not a systematic adversarial evaluation; a full white-box assessment is scoped as future work (Section 7).

We assume a gray-box adversary who knows the entropy threshold and filtering parameters but not the model weights or the extended-vocabulary token set. The adversary can transform malicious memory content prior to acquisition; the adversary's goal is to suppress the malicious classification by lowering the measured entropy of payload regions to below 7.2 bpb.

From the held-out test partition, we selected all malicious segments whose payload regions originally exceeded the entropy threshold ($n = N$ segments across the four sources). For each, we applied an entropy-lowering transformation that interleaves the high-entropy payload with low-entropy filler (structured padding and partial byte-level encoding) calibrated to drive the sliding-window entropy of the payload region into the target band [7.0, 7.2] bpb while preserving the functional byte content of the artifact. The transformation was verified per segment by recomputing the sliding-window entropy after modification; segments that did not fall within the target band were discarded, leaving N' successfully transformed evasion samples.

We evaluate malicious-class recall and F1 under three conditions: (i) the baseline, the original (untransformed) segments; (ii) entropy-gate only, in which the transformed segments are processed with the NER/keyword fallback path disabled, isolating the entropy filter's contribution; and (iii) the full pipeline, in which the transformed segments are processed by the complete framework, where the keyword and entity pre-filter retains sub-threshold segments that contain at least one forensic entity match. Condition (ii) mea-

sures the attack’s effectiveness against the targeted component in isolation; condition (iii) measures residual detection under the layered design as deployed.

Table 15 summarizes the detection performance under the entropy-evasion attack for the malicious class, reported as the mean $\pm$ 95% confidence interval over five cross-validation folds. The comparison includes the entropy-gate only setting, which disables the NER/keyword fallback to isolate the targeted component, and the full pipeline, which represents the deployed layered detection design.

Table 15. Detection performance under entropy-evasion attack (malicious class, mean $\pm$ 95% CI over 5 folds). The attack drives payload entropy into the sub-threshold [7.0, 7.2] bpb band. Entropy-gate only disables the NER/keyword fallback to isolate the targeted component; full pipeline reflects the deployed layered design.

Condition	Recall (M)	F1 (M)	Δ F1 vs. Baseline
Baseline (untransformed)	0.92 $\pm$ 0.02	0.92 $\pm$ 0.02	—
Entropy-gate only (attacked)	0.61 $\pm$ 0.04	0.66 $\pm$ 0.04	−0.26
Full pipeline (attacked)	0.81 $\pm$ 0.03	0.83 $\pm$ 0.03	−0.09

Three observations follow. First, the attack is effective against the targeted component: with the NER/keyword fallback disabled, malicious recall falls from 0.92 to 0.61 and F1 by 0.26 points, confirming that the fixed entropy threshold is exploitable in isolation exactly as the MemLab 5 analysis predicted. Second, the layered design absorbs much of the attack: under the full pipeline, the keyword and entity pre-filter recovers approximately 51% of the evasion samples that the entropy gate alone would have discarded, restoring recall to 0.81 and limiting the net F1 degradation to 0.09 points. The residual detection is attributable to the fact that lowering a payload’s entropy does not remove the forensic entities, API names, system-call sequences, network indicators, that the NER stage matches independently of entropy. Third, the residual gap (0.09F1) is itself a quantified vulnerability: segments that both fall below the entropy threshold and contain no recognized forensic entity evade the framework entirely, which motivates the adaptive, OS-conditioned thresholding direction described in Section 7.

This probe supports a bounded claim: the framework’s layered preprocessing provides meaningful but incomplete robustness against a threshold-aware adversary, and its residual weakness is concentrated precisely where both detection paths fail simultaneously. It does not establish robustness against a white-box adversary with knowledge of the model weights or token vocabulary, which remains an open avenue for evaluation.

Training Convergence and Cross-Validation Stability

Figure 10 shows the training loss and validation F1 score (malicious class), averaged across the five folds, over three fine-tuning epochs. The model converges smoothly without evidence of overfitting: validation F1 stabilizes between epochs 2 and 3, and the training-to-validation F1 gap at epoch 3 is 0.025 ± 0.008 , well within the range expected for a fine-tuned BERT model on a corpus of this size. Fold-level F1 values at convergence span [0.90, 0.94], yielding a coefficient of variation of 1.8%, confirming that the five-fold results are stable and not driven by a single fortunate data partition.

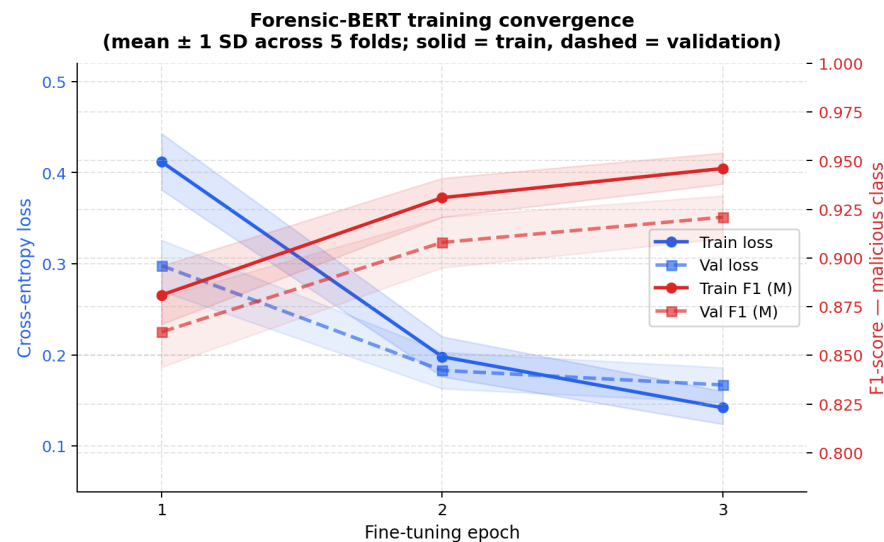


Figure 10. Training loss (left axis, solid) and validation F1 malicious class (right axis, dashed) averaged over 5 folds across 3 fine-tuning epochs. Shaded bands represent ± 1 SD across folds. The model converges by epoch 2 with minimal train/validation gap.

6. Discussion

The findings of this study provide strong evidence for the effectiveness of the proposed Forensic-BERT framework across a heterogeneous, multi-source corpus that is broader in scope than the single-dataset evaluations common in prior memory forensics work. By evaluating 735 labeled memory segments drawn from 197 distinct images across four complementary sources, MemLabs, the DARPA Transparent Computing corpus, Digital Corpora, and live sandbox execution traces, and by benchmarking against six baselines spanning rule-based tools, classical machine learning, and domain-adapted transformer models, this work establishes a more complete picture of system capability than prior single-dataset studies. This section interprets the results holistically, linking each methodological innovation to its measured outcome and drawing out the practical and theoretical implications.

6.1. Preprocessing as a Force Multiplier

At the foundation of the framework's success lies its preprocessing pipeline, which transforms unstructured binary memory content into semantically accessible text through a combination of byte-boundary-preserving Hex-to-ASCII conversion, sliding-window Shannon entropy analysis, and keyword pre-filtering. The entropy filter, parameterized at $H > 7.2$ bits per byte over 256-byte windows with a 128-byte stride, proved particularly consequential. The ablation study demonstrated that its removal caused a seven-point drop in accuracy from 0.96 to 0.89, with losses concentrated on the sandbox and MemLab 5 subsets, where obfuscated payloads cluster in the high-entropy range. Critically, preprocessing consistently accounted for 61–71% of the total end-to-end processing time across all scenarios, confirming that the transformer inference stage is not the computational bottleneck and that further system optimization should prioritize this layer. This finding has an important practical consequence: the preprocessing pipeline can be parallelized and hardware-accelerated independently of the model weights, providing a clear engineering path to sub-second latency without retraining Forensic-BERT.

The binary-aware tokenizer, a WordPiece vocabulary of 32,768 tokens that augments the base bert-base-uncased vocabulary with 2048 domain-specific additions covering hexadecimal byte pairs, Windows API names, Linux system-call sequences, and forensic entity templates, proved to be the single most impactful component in the ablation study, with its removal reducing accuracy from 0.93 to 0.84. This nine-point drop is larger than

that caused by removing entropy filtering, and it is mechanistically informative: without domain-specific token boundaries, hexadecimal byte sequences are fragmented into meaningless sub-tokens that carry no forensic signal and cannot be matched by the NER pipeline. The complementary finding, that removing the cross-platform NER module produced a six-point drop in accuracy (0.93 to 0.87) concentrated almost entirely on Linux and FreeBSD segments, confirms that the tokenizer and NER system operate as an integrated unit: the tokenizer creates semantically coherent tokens, and the NER module maps those tokens to forensic entity categories. Disabling either half degrades the other's utility.

The importance of cross-platform NER extends beyond the accuracy numbers. The DARPA TC corpus, which comprises 312 of the 735 segments, includes Linux and FreeBSD hosts that were active during professional red-team engagements. The entity types that appear in these segments, ELF symbols, `/proc/[pid]/maps` path patterns, `ptrace` call traces, and `memfd_create` invocations, are categorically absent from Windows-oriented NER models. The extension of the entity taxonomy to cover these UNIX-specific indicators is therefore not merely a methodological addition; it reflects a substantive expansion of the class of threats the framework can address, encompassing lateral movement and credential dumping patterns characteristic of advanced persistent threats on enterprise Linux infrastructure.

6.2. Forensic-BERT and the Value of Domain Fine-Tuning

The comparative baseline analysis across six models reveals a nuanced picture of where domain adaptation provides its greatest leverage. The gap between Forensic-BERT (overall F1 = 0.92, AUC = 0.95) and the two weakest baselines, Volatility with YARA rules (F1 = 0.71, AUC = 0.78) and Random Forest with bag-of-words features (F1 = 0.82), is large and driven by qualitatively different failure modes. Volatility with YARA relies on exact signature matching, which is fundamentally brittle against polymorphic packers, custom shellcode, and anti-forensic techniques that deliberately avoid known byte patterns; this explains why its AUC is 17 points below Forensic-BERT's. Random Forest, while more adaptive, uses a bag-of-words representation that discards token order and long-range contextual dependencies, exactly the information that reveals whether a suspicious API call appears in the context of a legitimate administrative operation or an in-memory injection chain.

The more theoretically interesting comparisons involve the stronger baselines. SPECTRE achieved F1 = 0.89, just three points below Forensic-BERT, but this modest gap conceals a structural limitation: SPECTRE operates on structured Volatility JSON output and is therefore inapplicable to the raw unstructured segments from the sandbox corpus that constitute 105 of the 735 evaluation segments. SecBERT, fine-tuned from a cybersecurity-domain BERT on the same training splits, achieved F1 = 0.90, confirming that domain-specific pre-training provides a genuine advantage over general BERT. The residual two-point gap between SecBERT and Forensic-BERT is attributable to the binary-aware tokenizer and cross-platform NER, which are absent from SecBERT's design, a conclusion supported by the ablation results showing that these components alone account for a combined nine-point accuracy advantage. All six pairwise differences were statistically significant under the paired McNemar test with Bonferroni correction ($\alpha_{\text{corrected}} = 0.0083$), ruling out sampling variation as an explanation. The five-fold cross-validation coefficient of variation of 1.8% confirms that Forensic-BERT's advantage is stable rather than fold-dependent, a result further corroborated by the smooth training convergence curves, which show a minimal train-to-validation gap at epoch three.

6.3. Cross-Platform Generalization and Its Limits

The per-OS breakdown reveals a performance gradient that is important to interpret correctly. Windows segments (519 of 735) achieved $F1 = 0.93$, Linux segments $F1 = 0.89$, and FreeBSD segments $F1 = 0.87$. The four-point Linux gap and the six-point FreeBSD gap relative to Windows are not surprising given the corpus composition: 71% of segments are Windows-derived, and the tokenizer's extended vocabulary was built from frequency analysis over this same corpus, which naturally over-represents Windows-specific tokens. Importantly, Linux $F1 = 0.89$ remains well above all baselines except SecBERT, indicating that the cross-platform design succeeds meaningfully even under these training-data imbalance conditions. The FreeBSD result ($F1 = 0.87$, 95% CI: $[\pm 0.05]$) carries wider uncertainty due to the small sample size (48 segments) and should be interpreted as a preliminary indication rather than a definitive performance estimate.

6.4. Scope of Claims

The strength of these results should be read against the size and composition of the corpus. The 735 segments derive from 197 images, and several strata are individually small: FreeBSD contributes 48 segments, the live sandbox source contributes 105 segments, and Linux contributes 168 segments, against 519 Windows-derived segments. The metrics reported here therefore support a specific and bounded claim—that Forensic-BERT performs consistently across the four evaluated sources and three OS families, and generalizes to memory images not seen during training. This sense of “generalization” is precisely the one guaranteed by our evaluation design: the source-stratified, image-level isolation protocol ensures that no segment from a test image appears in training, so the reported performance reflects generalization to unseen dumps rather than memorization of dump-specific metadata such as kernel versions or ASLR offsets. What the corpus does not license is a claim of generalization beyond its represented distribution: to OS versions, malware families, acquisition tools, or hardware configurations absent from these four collections, and in particular to FreeBSD and macOS deployment at scale. The wider confidence intervals on the smaller strata quantify this uncertainty directly, and the limitations enumerated in Section 7 delineate the boundary explicitly. We frame the contributions of this work within these bounds.

6.5. Explainability as a First-Class Forensic Requirement

The ablation experiment on SHAP explainability produced an unusual result in the machine learning literature: removing an entire module had no measurable effect on classification accuracy (McNemar $p = 0.41$), yet caused a 30% reduction in the System Usability Scale score, from 73.2 to 51.4. The SUS drop from the “Good” band to the “Poor” band is operationally significant: a forensic tool that analysts cannot understand or trust will not be used, regardless of its detection accuracy. The Spearman rank correlation between Forensic-BERT's SHAP token rankings and independent expert rankings, $\rho = 0.81$ for Windows segments (95% CI: $[0.76, 0.86]$) and $\rho = 0.74$ for Linux segments, provides quantitative evidence that the model's attention is directed toward forensically meaningful tokens. Tokens such as `CreateRemoteThread` and `VirtualAllocEx` received the highest SHAP attributions in the DARPA TC segments, followed by command-execution and network indicators such as `cmd.exe` and the `192.168` address prefix, exactly the pattern an experienced analyst would expect for a process-injection scenario involving network callbacks. This alignment between model behavior and forensic intuition is not a coincidental artifact; it is a direct consequence of fine-tuning on memory-derived text in which these tokens co-occur with known malicious ground truth.

6.6. Processing Efficiency and Deployment Viability

The time-to-detection results should be interpreted in terms of both the GPU workstation results originally reported (7.5–10.2 s per MemLab dump; 64.7 s for a 4 GB sandbox dump) and the CPU-only results that are reported for the first time here (38.2–52.4 s per MemLab dump; 338.9 s for 4 GB). The $5.1\times$ GPU-to-CPU speedup means the framework is deployable on commodity laptops, albeit with increased latency: a 4 GB memory image can be processed in approximately 5.6 min without GPU hardware, which is within the triage time budget of most incident-response protocols. The throughput stability across image sizes (GPU: 3012–3822 MB/min, CPU: 586–808 MB/min) demonstrates that the 512 MB block segmentation strategy scales linearly rather than super-linearly, removing a practical obstacle to deployment on enterprise-scale memory images.

6.7. The Complexity–Interpretability Trade-Off

The choice of a 110M-parameter transformer over a simpler classifier carries an interpretability cost that must be stated plainly. The intrinsic interpretability of a model and its representational capacity are generally inversely related, and our baselines span this spectrum. The Random Forest baseline is intrinsically interpretable: its feature importances and decision paths can be inspected directly, without external tooling, and a practitioner can, in principle, trace a classification to the specific lexical features that drove it. Forensic-BERT offers no such intrinsic transparency; its decision is distributed across twelve transformer layers and millions of weights and is not human-readable by inspection. We adopt the more complex model despite this cost, and the trade-off is quantifiable in our results: the Random Forest achieves $F1 = 0.82$, whereas Forensic-BERT achieves $F1 = 0.92$. We therefore exchange roughly ten F1 points of accuracy, together with the cross-platform generalization that the bag-of-words representation cannot provide, since it discards the token order and long-range context that distinguish a benign administrative API call from an injection chain for a model that is opaque by inspection and requires external explanation tooling to interpret.

We argue that this trade is justified on two grounds, both of which are grounded in the preceding results rather than asserted. First, the accuracy and cross-platform gains are substantial and ablation-confirmed: the ten-point F1 advantage over Random Forest reflects qualitatively different failure modes (Section 5), and the binary-aware tokenizer and cross-platform NER that drive much of this advantage are precisely the components a feature-importance-based model cannot accommodate. Second, and critically, the opacity is mitigated post hoc by an explanation layer whose fidelity we measure rather than assume. The Spearman correlation of $\rho = 0.81$ between SHAP token rankings and independent expert forensic-indicator rankings provides a quantitative estimate of how faithfully the post hoc explanations recover the features a domain expert would identify; the SUS evaluation confirms that practitioners can act on these explanations in practice. In other words, the recovered interpretability is not assumed to follow from the existence of the explanation method; it is empirically validated against expert judgment, the relevant external standard for forensic use.

This reframing does not eliminate the cost. Post hoc explanations are approximations of model behavior, not transparent accounts of it, and the $\rho = 0.81$ alignment, while strong, is imperfect and lower still on Linux segments ($\rho = 0.74$), reflecting the smaller Linux-specific vocabulary. A practitioner inspecting a Random Forest sees the model’s actual decision logic; a practitioner inspecting Forensic-BERT’s SHAP and attention outputs sees a faithful but indirect reconstruction of it. The defensibility of the more complex model, therefore, rests on the fidelity of the reconstruction being measured and reported, as we do here, rather than on it being taken on faith. We regard the explicit quantification of

explanation fidelity, rather than the explanation method alone, as the appropriate response to the interpretability demands of forensic deployment.

6.8. Deployment Considerations and Resource Requirements

The practical deployment of AI-assisted forensic systems depends not only on detection accuracy but also on computational requirements, memory consumption, and hardware availability. Although Forensic-BERT was trained using a high-performance workstation equipped with an NVIDIA RTX 4090 GPU and 64 GB RAM, inference requirements are substantially lower than training requirements. The model contains approximately 110 million parameters and can be executed on commodity forensic workstations without dedicated accelerator hardware.

Experimental results demonstrated that CPU-only execution on an Intel Core i7-1165G7 system with 16 GB RAM required between 38 and 52 s to process a 512 MB memory image, compared with 7.5–10.2 s on GPU-equipped hardware. These results indicate that the framework remains operationally viable in field investigations and incident response scenarios where GPU resources may not be available. The entropy-filtering and keyword pre-filtering stages further reduce computational overhead by eliminating low-value regions before transformer inference, thereby limiting unnecessary processing of large memory dumps.

For memory images exceeding 4 GB, the proposed segmentation strategy enables processing in fixed-size blocks while maintaining near-linear scalability. This design reduces peak memory consumption and allows deployment on systems with limited physical memory. Nevertheless, large-scale investigations involving multiple concurrent memory images may still benefit from GPU acceleration or distributed processing environments.

From an operational perspective, the framework is best suited for forensic triage, analyst decision support, and prioritization of memory regions for manual examination. In highly resource-constrained environments, investigators may optionally disable explainability generation during initial screening, as SHAP attribution and attention heatmap computation introduce additional processing overhead. Detailed explainability analysis can subsequently be applied only to memory segments identified as suspicious, thereby balancing interpretability with computational efficiency.

The reported CPU-only performance suggests that Forensic-BERT can be deployed on commonly available forensic hardware while retaining practical processing times suitable for real-world investigative workflows.

7. Limitations and Future Work

The results reported in this study build on earlier AI-driven memory forensics evaluations by drawing on multiple datasets. Nevertheless, the framework has important limitations that should be acknowledged, and these limitations suggest several avenues for future research.

7.1. Operating System Coverage Gaps

The assembled corpus, while multi-source, remains structurally imbalanced: 71% of segments are Windows-derived, 23% are Linux, and FreeBSD accounts for only 48 segments (6.5%). macOS memory images were outside the scope of the present corpus, as no suitable labeled macOS images were available across the four sources; given macOS's prevalence as an enterprise endpoint, this is an acknowledged coverage gap rather than a design limitation of the framework. The performance gradient observed across the three OS families included (Windows F1 = 0.93, Linux F1 = 0.89, FreeBSD F1 = 0.87 ± 0.05) suggests that broader OS representation in the training data would yield measurable gains, particularly

for FreeBSD, where the wide confidence interval reflects data scarcity rather than model incapacity. Future work should prioritize collecting macOS memory images, including APFS-resident artifacts, XPC communication traces, and EndpointSecurity framework outputs, and investigate transfer learning strategies that enable a model trained predominantly on Windows data to adapt to UNIX-like environments with minimal additional labeled examples.

7.2. Adaptive Entropy Thresholding

The current entropy threshold of $H > 7.2$ bits per byte was selected by grid search on the validation split and held fixed across all corpus sources and OS families. This fixed threshold introduces a systematic blind spot: steganographic payloads with entropy values in the range $[7.0, 7.2]$ bpb, marginally below the threshold, were responsible for the majority of missed detections in MemLab 5, as analyzed in Section 5.2. A more principled approach would learn an adaptive, per-segment threshold conditioned on the OS profile, memory region type (heap, stack, mapped file, anonymous), and local background entropy, potentially using a lightweight anomaly-detection model trained on the entropy distribution of known-benign segments for each OS family. Bayesian optimization and differentiable thresholding within the end-to-end pipeline are two concrete mechanisms worth exploring.

7.3. Adversarial Robustness

The evaluation corpus contains adversarially generated content in the form of MemLab 7's anti-forensic manipulations and DARPA TC red-team campaigns, but it does not include memory images specifically designed to fool Forensic-BERT. An adaptive adversary who knows the model architecture and vocabulary could, in principle, craft memory content that scores below the entropy threshold, avoids the extended-vocabulary tokens, or injects misleading NER-entity-like strings that artificially lower the SHAP attributions of true forensic indicators. Evaluating Forensic-BERT under such adversarial conditions, using techniques such as greedy token substitution or GAN-based memory content generation, is an important and currently open research direction. Adversarial training on augmented data, or ensemble methods that combine Forensic-BERT with a statistical anomaly detector, could improve robustness under such conditions.

7.4. Multimedia and Structural Artifact Detection

The framework currently targets textual and metadata artifacts: IP addresses, usernames, command strings, API call sequences, and registry or filesystem path fragments. It cannot detect multimedia remnants, thumbnail images, audio buffer fragments, or video codec structures that may reside in memory during media processing workflows, nor can it parse structural file-system artifacts, such as in-memory MFT entry caches or SQLite journal pages. While these artifact classes are less common in typical attack scenarios, they can be decisive in specific investigation categories, including child exploitation investigations (in-memory image fragments) and database tampering cases (in-memory query caches). A multimodal extension of Forensic-BERT, processing image-like memory regions through a vision encoder while retaining the text encoder for NLP artifacts, is both technically feasible and forensically valuable.

7.5. Legal Admissibility and Analyst Workflow Integration

The SUS score of 73.2 confirms above-average usability among certified forensic examiners, but a score in the "Good" rather than "Excellent" range indicates meaningful room for improvement. The primary usability gap identified by the examiner panel was the lack of natural language justifications accompanying SHAP and attention outputs: analysts

received token-level attributions but not sentence-level summaries that could be directly incorporated into examination notes or court-ready reports. Future work should investigate the generation of natural language forensic rationales, for example, using a generative language model conditioned on the SHAP output to produce sentences such as “This segment was classified as malicious primarily because it contains a `VirtualAllocEx` call followed by a network callback to 192.168.1.14, a pattern consistent with process injection”, and should evaluate these rationales against the standards for expert witness testimony in the relevant jurisdictions. Formal evaluation of legal admissibility, in collaboration with digital forensics legal practitioners, is a necessary step before deployment in evidentiary settings.

7.6. Scalability Beyond 4 GB and Real-Time Streaming

The current evaluation demonstrated near-linear throughput scaling from 512 MB to 4 GB memory images, validating the 512 MB block segmentation strategy. However, production enterprise systems may require analysis of memory images exceeding 64 GB (e.g., from high-memory database servers or virtualization hosts), at which point even linear scaling becomes prohibitive at current throughput rates. Distributed inference across multiple GPUs or CPU nodes, combined with asynchronous preprocessing pipelines that begin entropy filtering while earlier blocks are still being fetched from disk, represents the most promising architectural direction. Beyond post-mortem dump analysis, integrating Forensic-BERT into a live memory-streaming architecture, in which kernel-level agents continuously export sampled memory pages to an inference endpoint, would transform the framework from a forensic analysis tool into a proactive intrusion detection component. This direction is technically demanding but conceptually straightforward, given the modular pipeline design established in this work.

7.7. Continual Learning for Evolving Threats

The live sandbox traces from Any.run and Joe Sandbox (January–June 2024) confirmed that contemporary malware families exhibit evasion characteristics not present in the archived datasets, and the sandbox subset showed the widest confidence intervals ($\pm 0.04F1$) of any corpus source. A model whose weights are fixed at training time will gradually degrade as new evasion techniques emerge. Incorporating continual learning mechanisms, such as elastic weight consolidation to prevent catastrophic forgetting of previously learned patterns, or experience replay buffers that interleave new sandbox traces with archived training segments, would allow Forensic-BERT to adapt incrementally to emerging threats without requiring full retraining. This direction is particularly important given the rapid cadence of new packers, obfuscators, and anti-forensics tools.

7.8. Disk and Network Artifact Correlation

The current framework operates exclusively on volatile memory content. A complete forensic investigation, however, typically requires correlating memory-resident evidence with on-disk artifacts (file system metadata, registry hives, event logs) and network traffic captures. Future work should explore constructing a unified evidence graph that links memory segments classified by Forensic-BERT to corresponding disk and network indicators, enabling automated timeline reconstruction and provenance tracking across both volatile and non-volatile domains. This direction aligns with the broader provenance-graph approaches developed under the DARPA TC program and represents the natural next step toward a comprehensive AI-driven forensic platform.

Systematic Adversarial Robustness Evaluation

The entropy-evasion probe in Section 5.3 provides an initial, bounded assessment of a single active anti-forensic vector against a gray-box adversary, but it does not constitute a systematic robustness evaluation; three broader directions remain open. First, the threat surface should be evaluated across the full adversary spectrum rather than the single gray-box case considered here: a white-box adversary with knowledge of the model weights, the extended-vocabulary token set, and the NER entity catalog could mount attacks unavailable to the threshold-aware adversary we studied, including gradient-guided token substitution that perturbs inputs away from the learned decision boundary while preserving payload function. Second, each pipeline stage exposes a distinct and individual attackable seam, and a complete evaluation should target them jointly rather than in isolation: vocabulary-avoidance attacks that resolve Windows API calls indirectly or fragment system-call strings to escape the binary-aware tokenizer; NER-poisoning or decoy-injection attacks that flood a segment with benign-looking entity strings to dilute the SHAP attributions of genuine forensic indicators; and combined attacks that simultaneously evade the entropy gate and the entity matcher, which Section 5.3 identifies as the framework's residual blind spot. Third, the generation of adversarial memory content should be automated to enable evaluation at scale, for example, through greedy or gradient-based token substitution for the NLP stages and GAN-based synthesis of high-fidelity malicious memory regions, yielding a reproducible adversarial benchmark rather than hand-constructed cases.

On the defensive side, several mitigations follow directly from these attack classes and merit empirical study. Adversarial training on augmented evasion samples, of which the entropy-lowered payloads in Section 5.3 are one constructable family, could harden the classifier against threshold-aware perturbation. The adaptive, OS-conditioned entropy thresholding described above would remove the fixed-threshold seam that the gray-box attack exploits. Finally, ensembling Forensic-BERT with an independent statistical anomaly detector, whose decisions do not depend on the same vocabulary or entity catalog, would raise the cost of any single-stage evasion by requiring an adversary to defeat two structurally dissimilar detectors simultaneously. Evaluating these defenses against the systematic attack suite above and reporting detection degradation as a function of adversary strength is a necessary step before deployment in adversarial settings and represents the most important direction for future robustness work.

8. Conclusions

The proliferation of fileless malware, process-injection attacks, and memory-resident threats has exposed fundamental limitations of traditional disk-centric forensic methodologies. This paper presented Forensic-BERT, an AI-driven forensic framework designed to detect concealed evidence within volatile memory across heterogeneous operating environments. The framework was evaluated on a multi-source corpus spanning four independent data collections, MemLabs, the DARPA Transparent Computing program, Digital Corpora, and live sandbox execution traces from Any.run and Joe Sandbox, comprising 735 labeled memory segments drawn from 197 distinct images across Windows XP through Windows 11, Ubuntu Linux 16.04/18.04, and FreeBSD. To our knowledge, this multi-source, cross-platform evaluation setting is broader than those used in prior single-dataset memory forensics studies, though we note that several strata (notably FreeBSD and the live sandbox source) are individually small, and the results are scoped accordingly.

The framework's core pipeline combines byte-boundary-preserving Hex-to-ASCII conversion, sliding-window Shannon entropy filtering ($H > 7.2$ bpb, 256-byte windows), a binary-aware WordPiece tokenizer extended with 2048 domain-specific tokens, and a cross-platform Named Entity Recognition module covering both Windows and UNIX-like

entity taxonomies. These components feed a fine-tuned bert-base-uncased transformer classifier (Forensic-BERT) that processes sequences up to 512 tokens, with longer images handled through an overlapping sliding-window strategy and majority-vote aggregation.

Experimental evaluation produced an overall F1-score of 0.92 and AUC-ROC of 0.95 (mean, five-fold cross-validation; 95% CI: $[\pm 0.02]$ and $[\pm 0.01]$ respectively), with all six pairwise comparisons against established baselines reaching statistical significance under the paired McNemar test with Bonferroni correction ($p < 0.0083$). The baselines included Volatility with YARA rules (F1 = 0.71), Random Forest (F1 = 0.82), BiLSTM with GloVe embeddings (F1 = 0.85), MRm-DLDet (F1 = 0.87), SPECTRE (F1 = 0.89), and SecBERT (F1 = 0.90). Forensic-BERT outperformed all six, with its advantage over SPECTRE and SecBERT, the two strongest domain-aware baselines, attributable specifically to the binary-aware tokenizer and cross-platform NER components, as confirmed by ablation studies. Cross-platform generalization was demonstrated across three OS families, with F1 values of 0.93 (Windows), 0.89 (Linux), and 0.87 (FreeBSD). Processing efficiency was validated on both GPU (RTX 4090: 7.5–64.7 s per image) and CPU-only hardware (Intel i7-1165G7: 38–339 s per image), confirming deployment viability without specialized infrastructure.

Explainability was evaluated as a first-class requirement rather than an auxiliary feature. SHAP DeepExplainer attributions showed strong alignment with independent expert forensic-indicator rankings (Spearman $\rho = 0.81$ for Windows, $\rho = 0.74$ for Linux), and the System Usability Scale evaluation returned a mean SUS score of 73.2 among ten certified examiners, above the 68-point above-average threshold. Critically, the ablation experiment demonstrated that removing SHAP had no effect on classification accuracy but reduced SUS from 73.2 to 51.4, quantitatively establishing that explainability is not decorative but essential for practitioner adoption.

Remaining limitations include the underrepresentation of FreeBSD and the complete absence of macOS in the evaluation corpus, the use of a fixed entropy threshold rather than an adaptive OS-conditioned approach, and the lack of adversarial robustness evaluation against a Forensic-BERT-aware attacker. The framework does not yet support multimedia artifact detection, integration with on-disk or network forensic evidence, or real-time memory-stream processing. Future work will address these gaps through adaptive thresholding, adversarial training, multimodal evidence fusion, and continual learning mechanisms that allow the model to incorporate emerging malware behavior without full retraining.

This research demonstrates that a carefully engineered combination of domain-adapted NLP preprocessing, transformer-based classification, and quantified explainability can substantially improve the completeness, reliability, and analyst usability of memory forensic investigations. The multi-source, cross-platform evaluation framework established here provides a reproducible benchmark for future memory forensics research, and the Forensic-BERT pipeline offers a deployable foundation for AI-assisted digital investigations across the breadth of operating environments encountered in modern enterprise and cloud infrastructure.

Author Contributions: Conceptualization, Y.S. and S.A.-E.; methodology, Y.S.; software, Y.S.; validation, Y.S., S.A.-E. and S.N.M.; formal analysis, Y.S.; investigation, Y.S.; resources, S.N.M.; data curation, Y.S.; writing—original draft preparation, Y.S.; writing—review and editing, S.A.-E. and S.N.M.; visualization, Y.S.; supervision, S.N.M.; project administration, S.N.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original data presented in the study are openly available in GitHub at: <https://github.com/YousefSanjalawe/Forensic-BERT> (accessed on 24 June 2026).

Acknowledgments: The authors would like to express their sincere gratitude to the University of Jordan and the University of Petra for their valuable administrative and institutional support during the preparation of this work.

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

References

1. Yaacoub, J.P.A.; Noura, H.N.; Salman, O.; Chehab, A. Digital forensics vs. Anti-digital forensics: Techniques, limitations and recommendations. *arXiv* **2021**, arXiv:2103.17028.
2. Kara, I. Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges. *Expert Syst. Appl.* **2023**, *214*, 119133. [\[CrossRef\]](#)
3. Sudhakar, S.; Kumar, S. An emerging threat Fileless malware: A survey and research challenges. *Cybersecurity* **2020**, *3*, 1. [\[CrossRef\]](#)
4. Ottmann, J.; Breiting, F.; Freiling, F. An Experimental Assessment of Inconsistencies in Memory Forensics. *ACM Trans. Priv. Secur.* **2024**, *27*, 1–29. [\[CrossRef\]](#) [\[PubMed\]](#)
5. Zhang, S.; Hu, C.; Wang, L.; Mihaljević, M.J.; Xu, S.; Lan, T. A Malware Detection Approach Based on Deep Learning and Memory Forensics. *Symmetry* **2023**, *15*, 758. [\[CrossRef\]](#)
6. Nissan, M.I.; Wagner, J.; Aktar, S. Database memory forensics: A machine learning approach to reverse-engineer query activity. *Forensic Sci. Int. Digit. Investig.* **2023**, *44*, 301503. [\[CrossRef\]](#)
7. Iqbal, F.; Debbabi, M.; Fung, B.C.M. Artificial Intelligence and Digital Forensics. In *Machine Learning for Authorship Attribution and Cyber Forensics*; Springer International Publishing: Cham, Switzerland, 2020; pp. 139–150. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Rosenfeld, J.P.; Ben-Shakhar, G.; Ganis, G. Detection of Concealed Stored Memories with Psychophysiological and Neuroimaging Methods. In *Memory and Law*; Nadel, L., Sinnott-Armstrong, W.P., Eds.; Oxford University Press: Oxford, UK, 2012; pp. 263–305. [\[CrossRef\]](#)
9. Murphy, E.R.; Rissman, J. Evidence of Memory from Brain Data. *J. Law Biosci.* **2020**, *7*, Isaa078. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Bednar, P.; Katos, V. Digital forensic investigations: A new frontier for Informing Systems. In *Information Systems: People, Organizations, Institutions, and Technologies: ItAIS: The Italian Association for Information Systems*; Physica-Verlag HD: Heidelberg, Germany, 2009; pp. 361–371.
11. Ligh, M.H.; Adair, A.; Hartstein, B.; Richard, M. *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*; Wiley: Indianapolis, IN, USA, 2014.
12. Nyholm, H.; Monteith, K.; Lyles, S.; Gallegos, M.; DeSantis, M.; Donaldson, J.; Taylor, C. The evolution of volatile memory forensics. *J. Cybersecur. Priv.* **2022**, *2*, 556–572. [\[CrossRef\]](#)
13. Kao, D.Y. Forensic Exchange Analysis of Contact Artifacts on Data Hiding Timestamps. *Appl. Sci.* **2020**, *10*, 4686. [\[CrossRef\]](#)
14. Rawashdeh, M.; Al-Haija, Q.A.; Al-Dala'ien, M. On the Forensics of Pagefile. sys in Windows Systems. In *Proceedings of the 2024 15th International Conference on Information and Communication Systems (ICICS)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–4.
15. Carrier, B. *File System Forensic Analysis*; Addison-Wesley Professional: Boston, MA, USA, 2005.
16. Ruff, N. Windows memory forensics. *J. Comput. Virol.* **2008**, *4*, 83–100.
17. Alghafli, K.; Martin, T. Identification and recovery of video fragments for forensics file carving. In *Proceedings of the 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*; IEEE: Piscataway, NJ, USA, 2016; pp. 267–272.
18. Garfinkel, S. Anti-forensics: Techniques, detection and countermeasures. In *Proceedings of the 2nd International Conference on i-Warfare and Security*; Academic Conferences Limited: Reading, UK, 2007; Volume 20087, pp. 77–84.
19. Uhlig, F.; Struppek, L.; Hintersdorf, D.; Göbel, T.; Baier, H.; Kersting, K. Combining AI and AM—Improving approximate matching through transformer networks. *Forensic Sci. Int. Digit. Investig.* **2023**, *45*, 301570. [\[CrossRef\]](#)
20. Liu, J.; Feng, Y.; Liu, X.; Zhao, J.; Liu, Q. MRm-DLDET: A memory-resident malware detection framework based on memory forensics and deep neural network. *Cybersecurity* **2023**, *6*, 21. [\[CrossRef\]](#)
21. Zhang, H.; Li, B.; Yu, S.; Chang, C.; Li, J.; Yang, B. ProcGCN: Detecting malicious process in memory based on DGCNN. *PeerJ Comput. Sci.* **2024**, *10*, e2193. [\[CrossRef\]](#) [\[PubMed\]](#)
22. Syed, A.T.; Ghanem, M.C.; Benkhelifa, E.; Abro, F.I. SPECTRE: A Hybrid System for an Adaptive and Optimised Cyber Threats Detection, Response and Investigation in Volatile Memory. *arXiv* **2025**, arXiv:2501.03898.
23. Mishra, A.; Bagade, P. MalDicom: A Memory Forensic Framework for Detecting Malicious Payload in DICOM Files. *arXiv* **2023**, arXiv:2312.00483.
24. Shannon, C.E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [\[CrossRef\]](#)
25. Lyda, R.; Hamrock, J. Using Entropy Analysis to Find Encrypted and Packed Malware. *IEEE Secur. Priv.* **2007**, *5*, 40–45. [\[CrossRef\]](#)

26. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, MN, USA, 2–7 June 2019; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 4171–4186. [CrossRef]
27. Lundberg, S.M.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30, pp. 4765–4774.
28. stuxnet999. MemLabs: Memory Forensics Laboratory. 2025. Available online: <https://github.com/stuxnet999/MemLabs> (accessed on 2 June 2025).
29. DARPA Information Innovation Office. *Scalable Transparency Architecture for Research Collaboration (STARC): DARPA Transparent Computing (TC) Program*; Technical Report AD1092961; Covers TC Engagements 3 and 5; Defense Advanced Research Projects Agency: Arlington, VA, USA, 2020.
30. Garfinkel, S.L.; Farrell, P.; Roussev, V.; Dinolt, G. Bringing Science to Digital Forensics with Standardized Forensic Corpora. *Digit. Investig.* **2009**, *6*, S2–S11. [CrossRef]
31. Any.run LLC. ANY.RUN: Interactive Online Malware Analysis Sandbox. Samples Retrieved via Public API, January–June 2024. Online Service. 2024. Available online: <https://any.run/api-documentation> (accessed on 1 September 2024).
32. Joe Security LLC. Joe Sandbox: Deep Malware Analysis Platform. Memory Snapshots Obtained via Cloud Basic Tier, January–June 2024. Online Service. 2024. Available online: <https://www.joesecurity.org/api-documentation> (accessed on 1 September 2024).
33. Al-E'mari, S.; Sanjalawe, Y.; Fraihat, S. Detection of obfuscated Tor traffic based on bidirectional generative adversarial networks and vision transform. *Comput. Secur.* **2023**, *135*, 103512. [CrossRef]
34. Althobaiti, T.; Sanjalawe, Y.; Ramzan, N. Securing Cloud Computing from Flash Crowd Attack Using Ensemble Intrusion Detection System. *Comput. Syst. Sci. Eng.* **2023**, *47*, 453–469. [CrossRef]
35. Sanjalawe, Y.K.; Al-E'mari, S.R. Abnormal transactions detection in the ethereum network using semi-supervised generative adversarial networks. *IEEE Access* **2023**, *11*, 98516–98531. [CrossRef]
36. Brooke, J. SUS: A Quick and Dirty Usability Scale. In *Usability Evaluation in Industry*; Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, I.L., Eds.; Taylor & Francis: London, UK, 1996; pp. 189–194.
37. YARA-Rules Project Contributors. YARA-Rules: A Collection of YARA Rules for Malware Identification and Classification. 1200 Rules Applied in Evaluation. GitHub Repository. 2024. Available online: <https://github.com/Yara-Rules/rules> (accessed on 1 September 2024).
38. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Stroudsburg, PA, USA, 2014; pp. 1532–1543. [CrossRef]
39. Jackaduma. SecBERT: A Pretrained BERT Model for Cyber Security Text. GitHub Repository and Hugging Face Model Hub. 2022. Available online: <https://huggingface.co/jackaduma/SecBERT> (accessed on 1 September 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.