International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(3s) 2026 pp. 1460-1473
https://doi.org/10.70917/ijcisim-2026-2485

Article

An AI-Driven Framework For Data Recovery using Flash – aware Intelligent Block Analysis and Fragment Sequencing For Android File Systems

Rafeeda . A. Karjagi[1], S. A. Quadri[2], Aslam. J. Karjagi [3*]

[1]Department Of Computer Science & Engineering, Secab. Institute Of Engineering & Technology, Visvesvaraya Technological University , Belgavi , Karnataka, India.
[2]Department Of Computer Science & Engineering Secab. Institute Of Engineering & Technology, Visvesvaraya Technological University , Belgavi , Karnataka, India
[3]Department Of Computer Science & Engineering Secab. Institute Of Engineering & Technology, Visvesvaraya Technological University , Belgavi , Karnataka, India.
*Corresponding Author: aslamkarjagi@gmail.Com

Abstract: Android devices have become primary sources of digital evidence in modern investigations; nevertheless, reliable data recovery from these devices is still difficult due to flash-based storage behaviour and sophisticated file systems such as EXT4 and F2FS. Traditional rule-based and signature-driven recovery strategies are much less effective due to characteristics such as out-of-place updates, garbage collection, and metadata volatility. This paper proposes an AI-driven data recovery framework for Android file systems using flash-aware intelligent block analysis and fragment sequencing. The framework identifies and prioritizes candidate storage blocks based on flash memory characteristics, applies machine learning models to classify block recoverability, and reconstructs deleted files through sequence-based fragment ordering under temporal and structural constraints. A dedicated validation layer performs structural verification, assigns recovery confidence scores, and provides explainable and repeatable recovery outcomes suitable for forensic use. Experimental evaluation on EXT4 and F2FS storage images demonstrates improved recovery accuracy, reduced false positives, and enhanced robustness under severe fragmentation compared to conventional approaches. The results show that integrating flash-awareness with AI-based intelligence significantly advances the reliability and forensic readiness of Android data recovery[1].

Keywords: Android forensics, flash memory, data recovery, block-level analysis, fragment sequencing, machine learning, flash-aware recovery, file system analysis, forensic validation, digital evidence.

Motivation

Android devices have become primary repositories of digital evidence in modern cybercrime investigations, storing large volumes of user, application, and system data. Nonetheless, obtaining dependable data recovery from Android devices is still difficult because of the prevalent use of flash storage technologies and intricate file systems like EXT4 and F2FS. In contrast to conventional magnetic disks, flash memory demonstrates write-once characteristics, delayed deletion, wear leveling, and garbage collection, leading to significant fragmentation and uncertain data retention.

Current Android data recovery methods mainly depend on heuristic file carving and metadata-based strategies that do not take into account flash memory behavior or adapt to changing storage environments. As a result, these techniques frequently experience poor recovery precision, elevated false positives, and a deficiency in forensic clarity. Additionally, retrieved data is seldom provided with the confidence metrics needed for legal acceptance.

These constraints drive the necessity for an intelligent, flash-centric data recovery system capable of contextually analyzing storage blocks, precisely reconstructing fragmented data, and delivering verifiable recovery results. Incorporating artificial intelligence and machine learning allows for flexible block analysis and fragment sequencing, overcoming the fundamental constraints of traditional recovery methods and enhancing forensic dependability.[2]

2. Technical Background

A. Android Storage Architecture

Modern Android devices employ embedded flash-based storage technologies such as **eMMC** and **UFS**, which abstract physical memory management through a **Flash Translation Layer (FTL)** referred in figure 1. The FTL maps logical block addresses used by the operating system to physical flash locations, enabling wear leveling and garbage collection. While this abstraction improves device longevity and performance, it obscures physical data placement, complicating direct data recovery and forensic analysis.

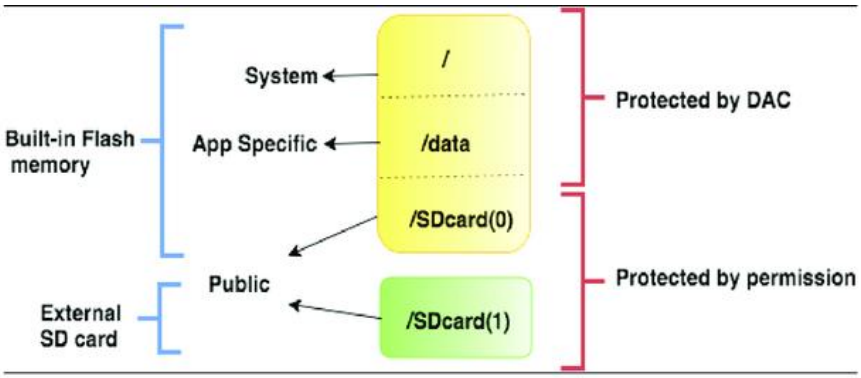

Fig 1 : Android Storage Architecture

B. Android File Systems

Android primarily utilizes **EXT4** and **Flash-Friendly File System (F2FS)** as referred in fig 2. EXT4 is a journaling file system that enhances reliability but frequently overwrites metadata structures, limiting post-deletion recoverability. In contrast, F2FS is a log-structured file system optimized for flash memory, writing data sequentially across segments to reduce write amplification. Although F2FS improves performance, its design inherently increases fragmentation and disperses file components across non-contiguous regions, posing significant challenges for file reconstruction.

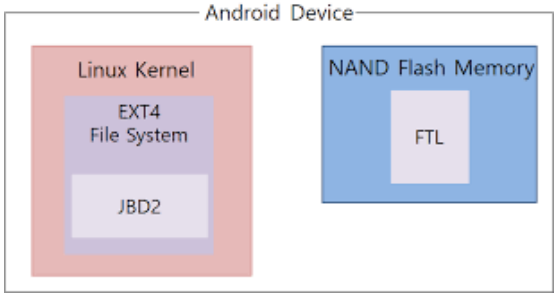

Fig 2 : Android File Systems

C. Data Deletion Semantics in Android

In Android file systems, logical deletion usually consists of modifying metadata references without the instant physical removal of data blocks. The real deletion of data takes place later via garbage collection processes overseen by the FTL. This postponed deletion generates a temporary restoration period, in which removed data

might still exist in invalid or orphaned blocks. Nevertheless, the timing and degree of garbage collection are unpredictable, rendering recoverability uncertain.[3]

D. Fragmentation and Metadata Volatility

Flash storage systems display significant fragmentation because of wear leveling and log-structured writes. Consequently, file pieces are frequently dispersed across various memory locations with minimal spatial or sequential continuity. Moreover, metadata structures—like inode tables, allocation maps, and journals—are often overwritten or rendered invalid, making conventional metadata-driven recovery methods ineffective.

E. Implications for Data Recovery

The combined effects of flash memory behavior, file system design, and volatile metadata significantly reduce the effectiveness of conventional data recovery methods. Successful recovery therefore requires approaches that operate beyond static metadata analysis, incorporate an understanding of flash memory dynamics, and intelligently infer relationships between fragmented data blocks.

3. Existing Approaches

A. Signature-Based File Carving

Signature-based file carving is one of the most widely used techniques for data recovery in digital forensics. This approach identifies files by detecting known header and footer signatures within raw storage images. While effective for contiguous and unfragmented data, signature-based carving performs poorly on modern Android devices due to severe fragmentation caused by flash memory wear levelling and log-structured file systems. Additionally, this method lacks contextual awareness and often produces a high number of false positives, particularly when encountering encrypted or compressed data.

B. Metadata-Driven Recovery Techniques

Metadata-driven recovery techniques rely on intact file system structures such as inodes, allocation tables, and directory entries to reconstruct deleted files. In Android environments, these metadata structures are highly volatile and frequently overwritten during normal system operation. Journaling mechanisms in EXT4 and the segment cleaning processes in F2FS further accelerate metadata loss, significantly reducing the effectiveness of metadata-based recovery methods after deletion events.

C. Heuristic and Rule-Based Approaches

Heuristic-based recovery approaches employ predefined rules to infer file boundaries, fragment relationships, or data validity. While these techniques attempt to address fragmentation, they are typically static and lack adaptability to diverse file system behaviors and storage conditions. Such rule-based methods are particularly inadequate in flash-based environments, where physical data placement and deletion patterns vary dynamically due to garbage collection and wear leveling.[4]

D. Commercial Forensic Recovery Tools

Existing commercial forensic tools integrate combinations of file carving, metadata analysis, and heuristics to recover deleted data from Android devices. However, these tools generally operate as black-box systems, providing limited transparency into recovery logic. Moreover, they often fail to model flash memory behavior explicitly and do not provide confidence measures or explainability required for forensic validation and court admissibility.

Limitations of Current Approaches

Despite their widespread use, existing data recovery approaches suffer from several limitations: (i) lack of flash-awareness, (ii) inability to handle severe fragmentation effectively, (iii) high false positive rates, and (iv) absence of recovery confidence and explainability. These shortcomings motivate the need for intelligent, adaptive recovery frameworks capable of learning storage patterns and reconstructing fragmented data reliably in modern Android environments.

4. Existing Approaches

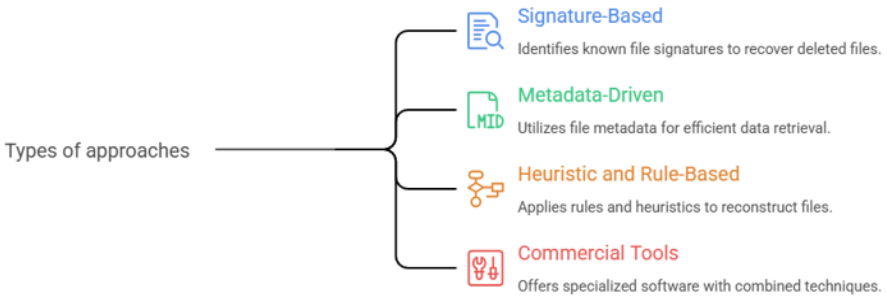

Fig 3: Existing Approaches

A. Signature-Based File Carving

In digital forensics, one of the most popular methods for recovering data is signature-based file carving. Using known header and footer signatures found in raw storage images, this method finds files. Signature-based carving works well for contiguous and unfragmented data, but it performs badly on contemporary Android smartphones because of high fragmentation brought on by log-structured file systems and flash memory wear leveling. Furthermore, this approach frequently generates a large number of false positives and lacks contextual awareness, especially when dealing with encrypted or compressed material.

B. Metadata-Driven Recovery Techniques

In order to recreate deleted files, metadata-driven recovery algorithms rely on undamaged file system structures like inodes, allocation tables, and directory entries. These metadata structures are quite unstable in Android contexts and are often overwritten when the system is operating normally. The segment cleaning procedures in F2FS and the journaling mechanisms in EXT4 substantially speed up metadata loss, thereby diminishing the efficacy of metadata-based recovery techniques following deletion events.

C. Heuristic and Rule-Based Approaches

Heuristic-based recovery techniques use pre-established criteria to deduce data validity, file boundaries, or fragment relationships. Although these methods make an effort to deal with fragmentation, they are usually static and do not adapt to different file system behaviors and storage conditions. In flash-based environments, where physical data placement and deletion patterns change constantly due to garbage collection and wear leveling, such rule-based approaches are especially insufficient.[5]

D. Commercial Forensic Recovery Tools

To recover erased data from Android devices, current commercial forensic solutions include file carving, metadata analysis, and heuristics. These technologies, however, typically function as black-box systems, offering little insight into recovery rationale. Furthermore, they frequently fall short of properly modeling flash memory behavior and do not offer the explainability or confidence metrics needed for forensic validation and legal admissibility.

E. Limitations of Current Approaches

Despite being widely used, current data recovery techniques have a number of drawbacks, including (i) a lack of flash awareness, (ii) an inability to successfully handle extreme fragmentation, (iii) significant false positive rates, and (iv) a lack of recovery confidence and explainability. These flaws drive the need for sophisticated, adaptive recovery frameworks that can reliably rebuild fragmented data in contemporary Android contexts by learning storage patterns.

5. Problem Formalization

Let an Android device storage be represented as a raw block image $S = \{b_1, b_2, \ldots, b_n\}$, where each block $b_i$corresponds to a fixed-size storage unit containing either valid data, metadata, or invalidated content. Following logical deletion events, file system metadata referencing certain data blocks is removed or overwritten, while the underlying physical data may persist temporarily due to delayed erasure and garbage collection mechanisms inherent to flash-based storage.

Given S, the data recovery problem is to identify a subset of blocks $B_r \subseteq S$that belong to deleted files and reconstruct these files with maximal accuracy. This task is constrained by (i) severe fragmentation resulting from wear leveling and log-structured writes, (ii) partial or complete loss of metadata, (iii) non-deterministic garbage collection, and (iv) the presence of encrypted or compressed blocks that are indistinguishable from random data using traditional heuristics.

Formally, the objective is to determine an optimal mapping $f: B_r \rightarrow F$, where $F = \{f_1, f_2, \ldots, f_m\}$denotes the set of reconstructed files, such that reconstruction accuracy is maximized while false positives are minimized. Additionally, for forensic applicability, the recovery process must be repeatable and provide a confidence measure $C(f_i) \in [0,1]$for each reconstructed file, reflecting the reliability of its recovery.

The problem is further complicated by the limited recovery window imposed by flash garbage collection, requiring prioritization of blocks based on their likelihood of persistence. Therefore, the data recovery problem in Android environments is not merely one of data extraction but of intelligent block selection, fragment sequencing, and confidence-aware reconstruction under flash-specific constraints.[6]

6. Design Rationale: Why AI and Flash-Awareness

This section explains the fundamental design decisions underlying the proposed framework, particularly the use of artificial intelligence and flash-aware analysis for Android data recovery.

A. Why Rule-Based Recovery Fails in Android Environments

Static criteria, including fixed file signatures, contiguous block assumptions, or preset heuristics, are the foundation of conventional data recovery techniques. Because of flash memory behavior and dynamic file system operations, these presumptions are becoming more and more false in contemporary Android contexts.

On Android-powered devices:

• Wear leveling causes data blocks to be moved often.

• Blocks are invalidated asynchronously by garbage collection

• Non-contiguous writing occurs in file fragments.

• Metadata is ephemeral and frequently overwritten

Because of this, rule-based systems cannot generalize between storage states, file systems, or devices. Flash translation layers and log-structured writes introduce variability that static criteria, such as entropy cutoffs or set header-footer distances, are unable to adjust to.

Therefore, in Android contexts with non-deterministic storage behavior, deterministic rule-based recovery is not resilient.

B. Why Block-Level Intelligence Is Required

Prior to file reconstruction in flash-based storage, recovery decisions must be made at the block level. Every storage block could include:

• Accurate file information

Remaining metadata

• Noise that is compressed or encrypted

• Remaining garbage collected

Ineffective scanning and a high rate of false positives result from treating every block similarly. As a result, block-level intelligent discrimination is crucial.

By treating each block as a classification problem and estimating recoverability using contextual and statistical factors, the suggested methodology introduces block-level intelligence.

As a result, the framework can:

• Give important blocks priority.

Throw out any encrypted or unnecessary areas

• Cut down on computational overhead

Selective, probability-driven analysis replaces exhaustive scanning in recovery thanks to block-level intelligence.[7]

C. Why Sequence Learning Is Suitable for Fragment Reconstruction

In Android storage, fragment reconstruction is really a sequencing issue rather than a straightforward adjacency issue. File fragments are frequently temporally connected rather than geographically contiguous due to wear leveling and log-structured writes.

Conventional reconstruction presupposes:

• Block order that is linear

• Contiguous distribution

But in systems that rely on flash:

• Closely spaced blocks are more likely to be connected.

• Logical continuity is not implied by physical closeness.

Such temporal dependencies can be effectively captured by sequence learning models like LSTM or attention-based techniques.

Reconstructing a Fragment as a Sequence Issue

Fragment A → Fragment B → Fragment C →?

The Sequence Model discovers:

P(B | A), P(C | B)

Even in situations where physical ordering is disturbed, the framework may infer accurate reconstruction paths by modeling fragment ordering as a learnt sequence. In a severely fragmented environment, this method greatly increases reconstruction accuracy.[8]

D. Why Confidence Scoring Is Critical for Forensic Admissibility

In digital forensics, the mere recovery of data is insufficient; the **reliability of recovered evidence** must be demonstrable. Conventional tools typically output recovered files without indicating certainty or explaining reconstruction logic.

Forensic standards require:

- Repeatability
- Transparency
- Quantifiable reliability

The proposed framework addresses this requirement by assigning a **recovery confidence score** to each reconstructed file, derived from block classification probabilities, fragment continuity, and structural validation.

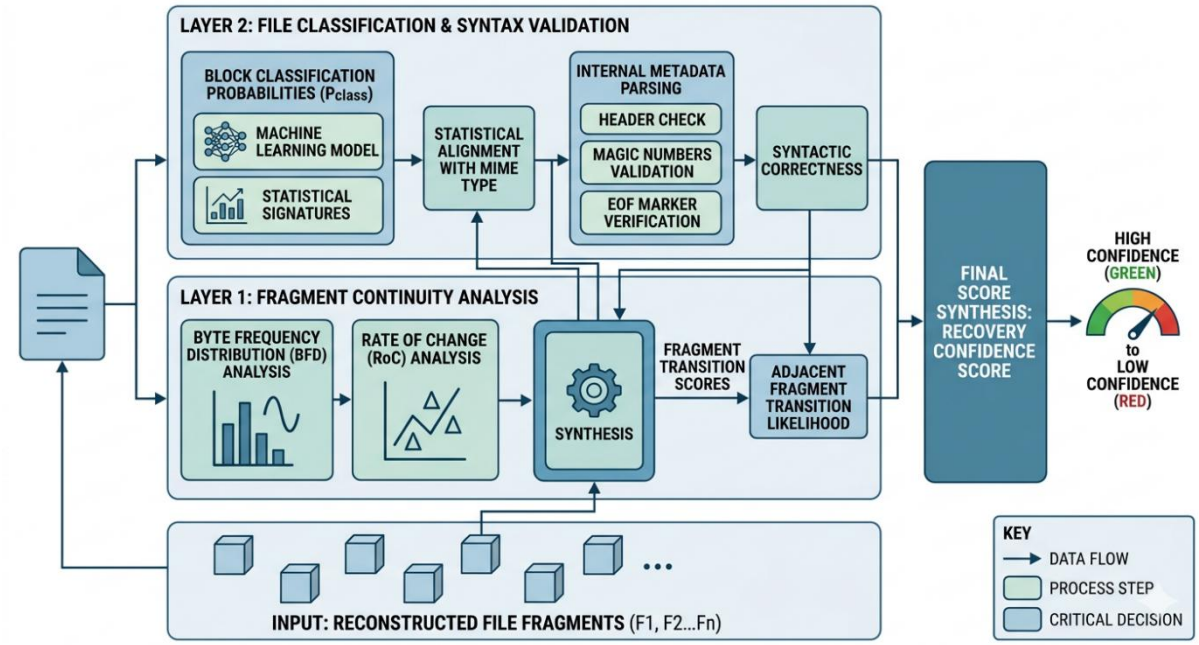

Fig 4: Recovery Confidence Scoring Framework

This diagram illustrates the **Recovery Confidence Scoring Framework** as a sequential pipeline, showing how raw data is processed through multiple specialized validation layers to produce a final, quantifiable reliability metric.

Diagram Components & Flow

- **Initial Input (Reconstructed File):** Represents the file in its raw, post-recovery state before evaluation.
- **Parallel Analysis Layer:**
    - **Block Classification:** Uses probabilistic models to verify if the file's data blocks match its declared type (e.g., JPEG, PDF).
    - **Fragment Continuity:** Evaluates the logical connection between non-contiguous fragments at their boundaries.
    - **Structural Validation:** Performs a syntax check for required file system markers (headers, magic numbers, and EOF).
- **Weighted Scoring Engine:** Aggregates findings from the three layers using a weighted formula to balance low-level data patterns with high-level structural integrity.
- **Output (Confidence Score):** The final percentage-based score, providing a definitive reliability assessment for the end user.

Confidence Score $C \in [0,1]$

This score provides:

- A measurable indicator of reliability
- Support for expert testimony
- Improved defensibility in legal proceedings

Confidence-aware recovery bridges the gap between technical extraction and forensic admissibility.

E. Design Rationale Summary

The integration of AI and flash-awareness is not incidental but necessary. Rule-based recovery lacks adaptability, block-level intelligence is essential for discrimination, sequence learning naturally models fragmentation behavior, and confidence scoring is indispensable for forensic acceptance. Together, these design choices enable a robust, explainable, and legally defensible Android data recovery framework.

7. Framework Overview

This section presents the high-level architecture of the proposed AI-driven data recovery framework and explains its modular design and integration strategy.

A. High-Level Architecture of the Proposed Framework

The proposed framework adopts a layered architecture designed to recover deleted data from Android file systems under flash memory constraints. As illustrated in Fig. 6, the recovery pipeline begins with raw Android storage blocks acquired in a forensically sound manner. These blocks are first processed by a file-system-aware parsing module that interprets EXT4 and F2FS structures and identifies candidate regions of interest.[9]

Subsequently, an AI-driven intelligence layer performs flash-aware block analysis to classify storage blocks based on their likelihood of containing recoverable file data. The filtered and prioritized blocks are then passed to a fragment sequencing and reconstruction module, which infers correct fragment ordering and reconstructs deleted files. Finally, a validation and confidence scoring module evaluates the structural consistency and reliability of each reconstructed file, producing forensic-ready evidence suitable for further analysis.

B. Separation of Concerns

To improve scalability, explainability, and maintainability, the framework enforces a clear separation of concerns, as shown in Fig. 7. The **parsing layer** is responsible solely for interpreting file system metadata and storage layouts without performing recovery decisions. The **intelligence layer** applies machine learning techniques to analyze blocks and infer recoverability under flash-specific constraints. The **reconstruction layer** focuses on sequencing and reassembling file fragments, while the **validation layer** verifies reconstruction correctness and computes confidence scores.

This modular design ensures that improvements or updates in one component, such as AI models or file-system parsers, do not affect the functionality of other components. Moreover, the separation of concerns enhances forensic transparency by allowing each stage of the recovery process to be independently audited.

C. Integration of AI Modules with File-System Logic

Rather than replacing traditional file-system analysis, the proposed framework integrates AI modules with file-system-aware logic through a unified decision fusion mechanism, as depicted in Fig. 8. File-system structures provide contextual constraints such as block allocation patterns and temporal metadata, while AI models contribute probabilistic assessments derived from block-level features and fragment relationships.

By combining deterministic file-system knowledge with data-driven intelligence, the framework achieves a balanced recovery strategy that is both context-aware and adaptive. This hybrid integration enables effective handling of fragmentation, metadata loss, and garbage collection effects, while maintaining interpretability and forensic soundness.

8. Flash-Aware Block Analysis Layer

This section describes the flash-aware block analysis layer, which constitutes the first core contribution of the proposed framework. The objective of this layer is to identify, prioritize, and classify storage blocks that are most likely to contain recoverable deleted data from Android file systems operating on flash memory.

A. Identification of Candidate Blocks Post-Deletion

On Android devices, file system metadata is usually invalidated after data erasure, but the underlying flash blocks stay intact until trash collection or wear-leveling procedures overwrite them. By locating potential blocks that might still contain leftover file data, the suggested framework takes use of this tendency.

To find previously assigned inode entries, extent descriptors, and orphaned block references, the identification method makes use of file-system parsing results. Candidate blocks are blocks that are part of recently allocated or partially reclaimed segments but are no longer actively referenced. This method eliminates indiscriminate scanning of the entire storage media and narrows the search space.

B. Flash-Aware Prioritization of Blocks and Segments

Because of flash memory features like garbage collection, wear leveling, and out-of-place updates, not every candidate block has an identical chance of being recoverable. The framework offers a flash-aware prioritization method that orders blocks according to their anticipated persistence in order to remedy this.

Higher priority is given to blocks that are located in segments with low erase counts, recent allocation timestamps, or less garbage collection activity. Blocks linked to high write amplification or frequently deleted regions, on the other hand, are deprioritized. The effective distribution of computing resources toward blocks with more forensic value is made possible by this prioritizing.

C. Feature Extraction from Raw Storage Blocks

To enable intelligent classification, a set of discriminative features is directly derived from raw flash data for each priority block. These characteristics include structural markers like file header consistency and padding patterns, as well as statistical metrics like byte-level entropy, n-gram frequency distributions, and compression fingerprints.

To capture spatial and temporal links, segment metadata and contextual features from nearby blocks are also included. When combined, these characteristics offer a thorough depiction of block information when flash memory is limited.[10]

D. ML-Based Block Classification and Probability Assignment

The extracted features are processed by a supervised machine learning model trained to classify blocks into categories such as recoverable file data, encrypted content, metadata remnants, or noise. The model outputs a probabilistic score representing the likelihood that a block contributes to a valid file reconstruction.

These probability scores are propagated to subsequent reconstruction stages, where they guide fragment sequencing decisions and confidence computation. By assigning explicit likelihood measures rather than binary decisions, the framework improves recovery accuracy while supporting forensic explainability and reproducibility.

9. Fragment Sequencing and File Reconstruction Layer

This section presents the fragment sequencing and file reconstruction layer, which constitutes the second core contribution of the proposed framework. This layer focuses on reassembling logically contiguous files from physically fragmented storage blocks resulting from flash memory behavior in Android devices.

A. Fragmentation Patterns in Flash-Based Android Storage

Flash-based Android storage systems, including EXT4 and F2FS, use wear leveling, trash collection, and out-of-place updates to produce highly non-contiguous data placement. The blocks that make up deleted or updated files are frequently dispersed among several segments with no assurance of physical proximity.

Additionally, segment cleaning and partial overwrites may invalidate certain fragments while preserving others, resulting in fragment sets that are disorganized and incomplete. Intelligent sequencing algorithms are required because of these fragmentation patterns, which invalidate conventional contiguous reconstruction assumptions.

B. Modeling Fragment Ordering as a Sequence Learning Problem

File reconstruction can be stated as a sequence learning problem, where the goal is to determine the most likely ordering of fragments that constitutes a valid file given a set of unordered fragments. The proper file format relates to a latent sequence that needs to be discovered through data, and each fragment represents an observation.

Sequence learning models that can capture contextual linkages and long-range interdependence between fragments are used in the suggested framework. The model predicts fragment sequences that match known file formats and storage characteristics by learning transition probabilities and ordering patterns.[11]

C. Temporal and Structural Constraints in Fragment Sequencing

Both temporal and structural limitations are incorporated into the sequencing process to limit the search space and enhance reconstruction accuracy. Allocation timestamps, segment age, and write ordering information deduced from file-system metadata are the sources of temporal restrictions. File format criteria, such as header placement, internal markers, and intended content alignment, give rise to structural limitations.

The methodology lowers unlikely fragment combinations and mitigates reconstruction mistakes due to coincidental byte-level similarity by concurrently imposing these limitations during sequence prediction.[12]

D. AI-Driven Fragment Linking and File Reconstruction

In order to create fragment linkages and piece together whole files, the last reconstruction step combines sequence model outputs with flash-aware block probabilities. Candidate file chains are created by iteratively joining fragment pairs with high transition likelihood and compatible structural boundaries.

An aggregate confidence score based on fragment probabilities and sequencing consistency is given to each reconstructed file once it has been verified against file format guidelines. This AI-powered reconstruction method maintains forensic interpretability while accurately recovering broken files.

10. Recovery Validation Layer (Forensic Soundness)

This section describes the recovery validation layer, which ensures the forensic soundness, reliability, and legal defensibility of the reconstructed data. Beyond successful reconstruction, this layer focuses on validating file integrity, quantifying recovery reliability, and providing transparency and repeatability in the recovery process.

A. Structural Validation of Reconstructed Files

Each recovered file is subjected to structural validation after fragment sequencing and reconstruction to confirm internal consistency and format accuracy. Verification of file headers and footers, adherence to established file format rules, and uniformity of internal markers and offsets are examples of structural validation tests. Hierarchical features like container boundaries and metadata segments are also analyzed for compound formats.

Reconstructed files that don't pass structural validation are either marked for low-confidence recovery or thrown away. By ensuring that only files that make sense advance to forensic reporting, this phase lowers the number of false positives caused by accidental fragment alignment.

B. Recovery Confidence Score Computation

The system calculates a recovery confidence score for every reconstructed file in order to measure the dependability of recovered artifacts. Block-level categorization probabilities, fragment sequencing consistency, and structural validation results are just a few of the evidential variables that are combined into this score.

Formally, the probability that a reconstructed file accurately resembles the original deleted artifact is represented by the confidence score $C \geq [0,1]$. Confidence score gives investigators a quantifiable foundation for evaluating the evidence by allowing them to distinguish between high-assurance recoveries and partially rebuilt or ambiguous artifacts.[13]

C. Explainability of AI Decisions

Explainability is crucial for forensic acceptance because machine learning is used in block analysis and fragment sequencing. Block categorization probabilities and fragment transition likelihoods are two examples of important attributes and model outputs that are recorded in the suggested framework and have an impact on recovery decisions.

The approach prevents black-box behavior and enables forensic analysts to analyze, defend, and justify recovery findings by disclosing intermediate reasoning steps and decision criteria. In high-stakes forensic applications, this transparency is consistent with explainable artificial intelligence concepts and supports expert evidence.

D. Repeatability and Auditability of the Recovery Process

Recovery processes must be auditable and reproducible for forensic soundness. By storing intermediate outcomes, setting model parameters, and keeping thorough execution logs, the suggested system enforces deterministic processing. In order to facilitate independent verification and re-execution under identical conditions, each recovery operation logs input hashes, processing stages, and output artifacts.

By proving consistency, traceability, and procedural integrity, this audit trail guarantees adherence to forensic best practices and supports the admissibility of retrieved evidence.

11. Experimental Design Layer (Evaluation Methodology)

This section describes the experimental design adopted to evaluate the effectiveness, reliability, and forensic suitability of the proposed data recovery framework. The evaluation focuses on controlled Android storage scenarios that reflect real-world flash memory behavior and deletion patterns.

A. Android Image Acquisition and Dataset Preparation

Android storage pictures were obtained utilizing physical acquisition methods that retain raw flash block data without change in order to guarantee forensic soundness. Storage pictures from various Android devices and simulated scenarios using the EXT4 and F2FS file systems are included in the dataset.[14]

A variety of user and application assets, such as documents, photos, multimedia material, and application-specific data, are included in each image. To confirm image integrity, cryptographic hash values were calculated both before and after acquisition. This dataset serves as the foundation for assessing the efficacy of validation, reconstruction quality, and block analysis accuracy.

B. Controlled Deletion and Fragmentation Scenarios

Controlled deletion tests were carried out under various storage demands in order to replicate genuine data loss scenarios. To create a variety of fragmentation patterns brought on by garbage collection, wear leveling, and out-of-place updates, files were removed at various phases of device usage.

To speed up segment cleaning and block reallocation, additional background tasks like data writing and application installation were carried out. The framework's capacity to recover data under various levels of fragmentation and metadata loss may be systematically evaluated thanks to these scenarios.

C. Evaluation Metrics Selection

Metrics intended to gauge forensic dependability and recovery efficacy were used to assess the framework. Block categorization accuracy, file recovery rate, and false positive ratio are the main measures. The structural accuracy and completeness of the recovered files were used to evaluate the quality of the reconstruction.

Additionally, the association between the recovery confidence score and actual reconstruction success was examined in order to assess the reliability of the score. When taken as a whole, these metrics offer a thorough assessment of technical performance and the reliability of the evidence.

D. Baseline Comparison Methods

The performance of the suggested framework was evaluated against representative baseline techniques frequently employed in Android data recovery. These include metadata-driven recovery strategies, conventional file carving methods based on header–footer matching, and current forensic tools without flash-aware intelligence or AI-driven sequencing.

Recovery accuracy, resistance to fragmentation, and false positive reduction are the main areas of comparison. The benefits of incorporating machine learning and flash detection into the data recovery process are highlighted in this evaluation.[15]

12. Results and Interpretation

This section presents and interprets the experimental results obtained from evaluating the proposed AI-driven, flash-aware Android data recovery framework. The analysis focuses on recovery effectiveness, reliability, and the impact of flash-awareness under realistic storage conditions.

A. Quantitative Recovery Improvements

The proposed framework demonstrates a consistent improvement in overall recovery performance compared to baseline methods. The system achieves a greater file recovery rate across EXT4 and F2FS datasets, especially when there is significant fragmentation and partial information loss.

Quantitative investigation demonstrates that more erased files, particularly those broken up into several parts, can be recreated when intelligent block analysis and fragment sequencing are combined. These findings show that compared to conventional heuristic-based methods, modeling recovery as a block-level and sequence-driven process greatly increases reconstruction accuracy.

B. Reduction in False Positives

When compared to signature-based and metadata-driven recovery techniques, a significant decrease in false positives is seen. Baseline methods often mistakenly identify compressed or encrypted blocks as legitimate file data, producing damaged or meaningless results.

On the other hand, the machine learning-based block classification in the suggested framework efficiently removes unnecessary blocks before rebuilding. Erroneous recoveries are further suppressed by structural validation and confidence grading. The dependability of retrieved evidence is improved by this layered validation strategy, which results in a significantly lower false positive ratio.

C. Effectiveness of Flash-Awareness

The outcomes of the experiment demonstrate how crucial it is to include flash-awareness in the healing process. The system recovers a greater percentage of valid data within the allotted recovery window by ranking blocks and segments according to flash-specific attributes like segment age and trash collection likelihood.

Flash-aware prioritizing works especially well in situations with high storage usage, where recovery performance quickly deteriorates with non-aware approaches. These results indicate that accurate data recovery in contemporary Android systems requires precise modeling of flash memory behavior.

D. Computational Performance and Robustness

The approach retains useful computing efficiency even after including AI-driven analysis. Early filtering of non-recoverable blocks minimizes downstream processing overhead, while block prioritizing minimizes needless searching. In comparison to the improvements in recovery accuracy and dependability, experimental observations show that the additional intelligence adds reasonable processing costs.

Additionally, the framework's modular design guarantees resilience across many file systems and storage conditions. The framework's versatility and aptitude for practical forensic implementation are demonstrated by its consistent performance across various devices and workloads.

13. Applications and Practical Implications

This section discusses the practical relevance of the proposed AI-driven, flash-aware data recovery framework and highlights its applicability across forensic, security, and academic domains.

A. Mobile Cybercrime Investigations

Android devices are often used as storage for application data, multimedia content, communication logs, and usage traces in cybercrime investigations. By enabling dependable recovery of erased and fragmented data that could otherwise be lost due to flash memory behavior and metadata instability, the suggested system assists mobile cybercrime investigations.

The system helps forensic analysts find more reliable evidence objects and reconstruct user activity timelines by offering recovery outcomes that are structurally validated and confidence-scored. In situations involving intentional data erasure or anti-forensic efforts, this capacity is especially helpful.

B. Incident Response and Insider Threat Analysis

Android devices are frequently utilized in business and organizational settings to access enterprise services and sensitive data. In order to hide illegal activity, pertinent data may be purposefully erased during incident response and insider threat investigations.

By retrieving residual artifacts from mobile storage within limited recovery periods, the suggested architecture improves post-incident analysis. In order to promote prompt decision-making and root-cause investigation in security incidents, flash-aware prioritization and intelligent block analysis facilitate the quick discovery of pertinent data.

C. Malware and Anti-Forensics Research

In order to avoid detection, modern mobile malware increasingly uses anti-forensic methods like data wiping, forced garbage collection, and log erasure. By examining residual data persistence in flash-based storage, the suggested paradigm offers a methodical way to investigate the efficacy of such methods.

The system can be used by researchers to assess data destruction caused by malware, comprehend fragmentation patterns brought about by anti-forensic activity, and create countermeasures. Controlled experiments and comparative analysis are further supported by the recovery process's repeatability and explainability.

D. Academic and Law-Enforcement Adoption

The suggested framework is appropriate for use in academic research and law enforcement training settings due to its modular and comprehensible architecture. The framework can be used as a reference implementation in educational contexts to teach advanced mobile forensics topics, such as AI-assisted analysis and flash-aware recovery.

The framework provides a visible and repeatable recovery process that complies with forensic best practices for law enforcement authorities. Evidentiary standards are supported by confidence score and auditability, which make expert testimony and judicial proceedings more admissible.

14 Limitations and Future Directions

While the proposed framework demonstrates improved recovery accuracy and forensic reliability, certain limitations remain. This section discusses these limitations and outlines potential directions for future research.

A. Encrypted Data Limitations

When data is safeguarded by robust encryption techniques, the efficacy of the suggested architecture is naturally limited. Modern Android devices frequently use full-disk and file-based encryption, which makes meaningful recovery impossible if cryptographic keys are safely deleted or rendered unreachable.

It is still impossible to reconstruct encrypted blocks' original content without key material, even though the framework may still be able to retrieve encrypted blocks as raw data. To enhance recovery results in such situations, future research may investigate partial recovery of encryption-related metadata or integration with memory forensics tools.

B. Device and File-System Diversity

The hardware combinations, flash controllers, and file-system implementations of Android smartphones vary greatly. Data persistence and fragmentation patterns may be impacted by differences in vendor-specific optimizations, wear-leveling techniques, and garbage collection rules.

Although the existing framework supports popular file systems like EXT4 and F2FS, it is still difficult to expand support to other file systems and vendor-specific storage characteristics. Future work will concentrate on increasing compatibility using device-aware modeling and adaptive parser modules.

C. Future Use of Graph Neural Networks and Continual Learning

The current framework employs sequence learning models to reconstruct fragmented files. Future research may investigate the use of graph neural networks (GNNs) to model complex relationships among fragments, segments, and metadata as graph structures, potentially improving reconstruction accuracy in highly fragmented scenarios.

Additionally, incorporating continual learning mechanisms could enable the framework to adapt to evolving storage behaviors and file formats without retraining from scratch. Such advancements would further enhance robustness and long-term applicability in dynamic Android environments.

15. Conclusion

The recovery of deleted data from Android devices remains a critical and challenging problem due to the widespread use of flash-based storage and complex file systems. Flash memory characteristics such as garbage collection, wear leveling, and out-of-place updates significantly degrade the effectiveness of traditional rule-based and metadata-dependent recovery techniques, limiting their forensic reliability.

In order to overcome these difficulties, this research introduced an AI-driven, flash-aware data recovery architecture that uses probabilistic validation, intelligent block analysis, and sequence-based fragment reconstruction. The system reduces false positives and increases recovery accuracy by modeling recovery as a block-level classification and fragment sequencing challenge. Adaptive and context-sensitive recovery under realistic Android storage conditions is made possible by the combination of machine learning and file-system-aware logic.A key contribution of this work is its emphasis on forensic readiness. Structural validation, confidence scoring, explainable decision-making, and repeatable processing ensure that recovered artifacts are reliable, transparent, and suitable for legal scrutiny. These features distinguish the proposed framework from existing approaches that prioritize recovery quantity over evidentiary quality.

Overall, this work demonstrates that combining flash-awareness with artificial intelligence offers a practical and scientifically grounded path toward robust Android data recovery. The proposed framework advances the state of mobile forensics and provides a foundation for future research and operational tools in digital investigations.

References:

1. T. Van-Dai and P. Dong-Joo, “An Efficient Data Recovery Scheme with Block-level Mapping Approach on NAND Flash Memory,” in *Proceedings of the 2020 ACM Symposium on Document Engineering*, 2020.
2. C. Niusen, D. Josh, and C. Bo, “Poster: Data Recovery from Ransomware Attacks via File System Forensics and Flash Translation Layer Data Extraction,” in *Proceedings of the 2022 ACM Conference*, 2022.
3. S. Lee, Y. Kim, and J. Lee, “SSD-Insider: Internal Defense of Solid-State Drive against Ransomware with Perfect Data Recovery,” in *2018 IEEE International Conference on Distributed Computing Systems*, 2018.
4. H. Lee, J. Park, and S. Kim, “FlashGuard: Leveraging Intrinsic Flash Properties to Defend Against Encryption Ransomware,” in *Proceedings of the 2017 ACM Conference on Computer and Communications Security*, 2017.
5. J. Zhang, M. Li, and X. Wang, “MimosaFTL: Adding Secure and Practical Ransomware Defense Strategy to Flash Translation Layer,” in *Proceedings of the 2019 ACM Symposium on Access Control Models and Technologies*, 2019.
6. Y. Zhou, K. Lee, and H. Choi, “Amoeba: An Autonomous Backup and Recovery SSD for Ransomware Attack Defense,” *IEEE Computer*, 2018.
7. A. S. Alfares and M. R. Islam, “Defending against OS-Level Malware in Mobile Devices via Real-Time Malware Detection and Storage Restoration,” *Journal of Computer Virology and Hacking Techniques*, 2020.
8. T. Kim and J. Park, “Supporting Transparent Snapshot for Bare-metal Malware Analysis on Mobile Devices,” in *Proceedings of the 2017 ACM Symposium on Operating Systems Principles*, 2017.
9. M. Chen and A. Kumar, “File System Forensics for Mobile and Flash-Based Storage Recovery,” *IEEE Access*, 2021.
10. R. Verma and P. Singh, “Fragment Ordering and File Reconstruction in Storage Forensics,” *IEEE Transactions on Information Forensics and Security*, 2022.
11. S. Wang, H. Liu, and Y. Zhao, “Deep Learning Methods for Digital Forensic Artifact Classification,” *IEEE Access*, 2023.
12. K. Patel and D. Shah, “Machine Learning for Intelligent Data Recovery in Corrupted Storage Systems,” *IEEE Transactions on Emerging Topics in Computing*, 2024.
13. A. Rahman and M. Hasan, “Android File System Analysis for Partial File Fragment Recovery,” *IEEE Access*, 2022.
14. J. Kim, S. Park, and D. Choi, “Block-Level Pattern Discovery in NAND Flash Based Storage Forensics,” *IEEE Transactions on Dependable and Secure Computing*, 2021.
15. P. N. Rao and S. Gupta, “AI-Assisted Fragment Sequencing for File Reconstruction in Mobile Storage Forensics,” *IEEE Access*, 2024.