Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

SilentEye: A Multi-Stage Static Detection Framework for Malware-Embedded Multimedia Files Using Heuristic Analysis and Random Forest Steganalysis

Shreyas Santosh Walake
Department of Computer Engineering
JSPM's JSCOE, Pune

Prasad Vilas Wakchaure
Department of Computer Engineering
JSPM's JSCOE, Pune

Vaishnavi Sidram Shinde
Department of Computer Engineering
JSPM's JSCOE, Pune

Shravani Chittaranjan Takale
Department of Computer Engineering
JSPM's JSCOE, Pune

Prof. Shikha Dwivedi
Department of Computer Engineering
JSPM's JSCOE, Pune

Abstract - **Multimedia files distributed through messaging platforms such as WhatsApp and Instagram represent an increasingly exploited vector for malware delivery, wherein malicious payloads are concealed within images, video, audio, PDF, and document formats through steganographic embedding and container injection techniques. Conventional antivirus solutions primarily target executable file formats and demonstrate limited efficacy against threats embedded within media file structures. This paper presents SilentEye, an offline multi-stage static detection framework designed to identify malware-embedded multimedia files without dependency on cloud infrastructure or behavioral runtime analysis. The proposed system implements a three-stage pipeline: a lightweight preprocessing layer performing extension validation, magic byte verification, double-extension detection, and hash-based reputation lookup; a heuristic decision engine executing twenty-nine YARA rules alongside format-specific detectors covering overlay injection, polyglot file identification, PDF exploit signatures, Office macro detection, SVG script injection, and audio-visual container anomalies across six file categories; and a machine learning layer employing four specialized Random Forest classifiers for steganalytic detection across JPEG, PNG, video, and SVG formats. The heuristic engine was evaluated on a dataset of 3,585 files achieving a true positive rate of 93.8% and a false positive rate of 0.8% at an average scan latency of 166 milliseconds. The machine learning layer was evaluated on 36,800 files sourced from six benchmark datasets including ALASKA2 and BOSSBase, achieving F1 = 0.9965 on Steghide-generated JPEG samples, while achieving F1 = 0.5231 on ALASKA2 J-UNIWARD adaptive steganography. The remaining classifiers achieved F1 = 1.0 for PNG alpha-channel, AVI payload injection, and SVG script-injection detection. A smart-read memory architecture enables scanning of files up to 800 MB within a 4 MB RAM footprint, making the system deployable on consumer-grade hardware without GPU requirement. Limitations including JUNIWARD adaptive steganography detection failure and audio LSB analysis on music recordings are formally characterized and contextualized against published steganalysis benchmarks.**

Keywords - Steganography, Steganalysis, Malware Detection, Multimedia Security, Static Analysis, YARA Rules, Random Forest, Digital Forensics, Offline Security, Messaging Application Security, Media File Forensics, Anomaly Detection, Feature Engineering, Information Hiding, Cyber Threat Detection.

I. INTRODUCTION

Instant messaging platforms have become the dominant channel for file exchange across personal and professional networks, with WhatsApp and Instagram collectively serving over three billion active users worldwide. This scale of media sharing has not gone unnoticed by threat actors, who increasingly exploit multimedia files as delivery vehicles for malicious payloads. Unlike executable-based attacks, media-borne threats conceal malicious content within the structural internals of images, video containers, audio files, PDF documents, and office formats, making them largely invisible to conventional antivirus engines that remain optimized for executable file inspection [10].

Steganography has transitioned from an academic discipline into an operational attack technique. Documented campaigns embed payloads through JPEG quantization table manipulation [4], PNG alpha channel encoding, RIFF chunk boundary injection in AVI containers, and JavaScript execution within SVG files [11]. These vectors are particularly effective against users who implicitly trust media files received through messaging applications, as no executable extension is present to trigger behavioral suspicion.

Existing detection approaches suffer from three limitations that SilentEye is designed to address. Signature-based antivirus tools lack format-specific parsing depth for media container internals [14]. Academic steganalysis tools such as those built on rich models [13] and CNN architectures [5], [15] address single file formats and require GPU inference, making them unsuitable for consumer hardware deployment. Cloud-based scanning introduces unacceptable privacy risk for air-gapped and privacy-sensitive environments. No existing unified framework addresses all three constraints simultaneously across multiple media formats.

India represents the highest-priority deployment context for this work. As the largest WhatsApp user base globally, India has seen

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

documented campaigns exploiting media file delivery for UPI payment fraud and fake application distribution, as reported by CERT-In [11]. This threat context directly motivated the design decisions in SilentEye.

As shown in Fig. 1, SilentEye implements a four-component offline pipeline: a preprocessing layer for lightweight file validation and hash reputation lookup against MalwareBazaar; a statistical analysis engine executing twenty-nine YARA rules with format-specific heuristic detectors across six file categories; a machine learning layer with four specialized Random Forest classifiers [12] for steganalytic detection; and a hybrid fusion engine combining both analysis streams into a final verdict of clean, suspicious, or malicious with automatic quarantine on malicious classification.

The key contributions of this work are: a unified multi-format static detection pipeline requiring no cloud dependency or GPU; a smart-read architecture scanning files up to 800 MB within 4 MB RAM; a heuristic engine achieving 93.8% true positive rate and 0.8% false positive rate across 3,585 files; a Random Forest steganalysis layer achieving F1 of 0.9965 on JPEG steghide detection across 36,800 files from ALASKA2 [6] and BOSSBase [7]; and formal characterization of detection boundaries against JUNIWARD adaptive steganography [4].

II. RELATED WORK

A. Steganography and Steganalysis Techniques

Early steganalysis research established statistical methods for detecting LSB embedding in digital images. Westfeld and Pfitzmann [1] introduced chi-square analysis demonstrating that LSB substitution produces measurable uniformity in bit-pair frequencies, forming the theoretical basis for passive steganalysis. Fridrich et al. [2] extended this through Regular-Singular analysis, providing an estimator for LSB embedding rate that remains a standard benchmark in steganalysis literature. However, Ker [3] demonstrated that both approaches fail on natural camera images where sensor noise produces LSB distributions statistically indistinguishable from embedded content, a limitation directly encountered in SilentEye's audio LSB detection module.

The field shifted toward adaptive steganography algorithms designed to minimize embedding distortion. Holub et al. [4] introduced J-UNIWARD, which embeds payload by targeting DCT coefficients in complex image regions while avoiding smooth areas, producing statistical footprints that chi-square and RS analysis cannot reliably detect. This shift rendered handcrafted feature approaches insufficient for modern JPEG steganalysis, as confirmed by the random-chance performance of Random Forest baselines on ALASKA2 JUNIWARD reported by Cogranne et al. [6].

B. Deep Learning for Steganalysis

The inadequacy of handcrafted features against adaptive algorithms motivated a transition to learned representations. Qian et al. [17] demonstrated that convolutional neural networks can learn discriminative residual features directly from image data without manual feature engineering. Xu et al. [15] refined CNN architectural design for steganalysis through systematic structural analysis, while Ye et al. [16] introduced hierarchical feature representations achieving improved detection on adaptive algorithms. Yedroudj et al. [18] further advanced this direction with an efficient spatial domain CNN achieving competitive accuracy at reduced computational cost.

Boroumand et al. [5] established the current deep learning benchmark through SRNet, a deep residual architecture achieving 64.3% accuracy on ALASKA2 JUNIWARD detection. Yousfi et al. [13] subsequently demonstrated that transfer learning from pre-trained CNNs via EfficientNet-B4 achieves 68.1% accuracy on ALASKA2, representing the strongest published result on this benchmark. These results collectively establish that JUNIWARD detection requires CNN-based approaches with millions of parameters, which is beyond the scope of CPU-deployable systems targeting consumer hardware. SilentEye acknowledges this boundary explicitly and restricts its JPEG ML claims to steghide detection where Random Forest achieves F1 of 0.9965.

C. Malware Concealment in Media Files

Selvaraj et al. [14] surveyed the transition from classical machine learning to deep learning in image steganalysis, documenting how handcrafted features such as spatial rich models and histogram-based approaches are progressively outperformed by CNN architectures on adaptive algorithms while remaining effective against tool-based steganography such as steghide and OutGuess. Chaganti et al. [10] systematically characterized stegomalware as a distinct threat class, documenting real-world use of image steganography for malware delivery and noting the absence of unified detection frameworks capable of addressing multiple media formats simultaneously. Strachanski et al. [11] provided the most recent comprehensive taxonomy of stegomalware patterns including polymorphic payload embedding, container injection, and metadata exploitation across image, video, and document formats, directly informing the attack coverage scope of SilentEye Stage 2A.

D. Datasets and Benchmarks

Rigorous evaluation of steganalysis systems requires standardized datasets with controlled embedding conditions. The ALASKA2 challenge dataset introduced by Cogranne et al. [6] provides 20,000 JPEG images with multiple embedding algorithms at varying payloads, representing the most demanding public benchmark for JPEG steganalysis. BOSSBase introduced by Bas et al. [7] provides a standard corpus of 10,000 grayscale images widely used for spatial domain steganalysis evaluation. The UCF101 dataset by Soomro et al. [8] provides action recognition video content repurposed in this work for AVI container payload injection evaluation. The DIV2K dataset by Agustsson and Timofte [9] provides high-resolution PNG images used for alpha channel steganography evaluation. SilentEye's ML evaluation draws from all four datasets, totaling 36,800 files across six benchmark sources.

E. Positioning of SilentEye

Existing steganalysis research addresses individual file formats with format-specific models requiring GPU inference, while existing malware detection tools lack the format-specific parsing depth to inspect media container internals. SilentEye occupies the intersection of these domains, providing a unified static analysis framework across six file categories deployable on consumer CPU hardware without cloud dependency. To the best of our knowledge, we are unaware of a publicly documented framework that combines heuristic media-container analysis and Random Forest-based steganalysis across image, video, audio, PDF, document, and SVG formats within a single offline pipeline.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

III. SYSTEM ARCHITECTURE

A. Overview

SilentEye is designed as a fully offline, multi-format static malware detection framework operating without cloud dependency, behavioral runtime analysis, or GPU requirement. The system processes multimedia files received through messaging platforms and produces a three-class verdict of clean, suspicious, or malicious across six file categories: image, video, audio, PDF, office document, and SVG. As illustrated in Fig. 1, the pipeline comprises four sequential components — preprocessing, statistical analysis, machine learning inference, and hybrid fusion — each contributing distinct detection capability while sharing a unified file representation to eliminate redundant processing overhead.

The architectural novelty of SilentEye lies in three design decisions absent from prior work. First, a smart-read memory model reads only the header region and a dynamically computed tail region of each file, enabling scanning of files up to 800 MB within a 4 MB RAM footprint without loading full file content into memory. Second, artifact-tagged score deduplication prevents inflation when multiple detection layers identify the same underlying artifact, ensuring that a single embedded PE header does not accumulate disproportionate

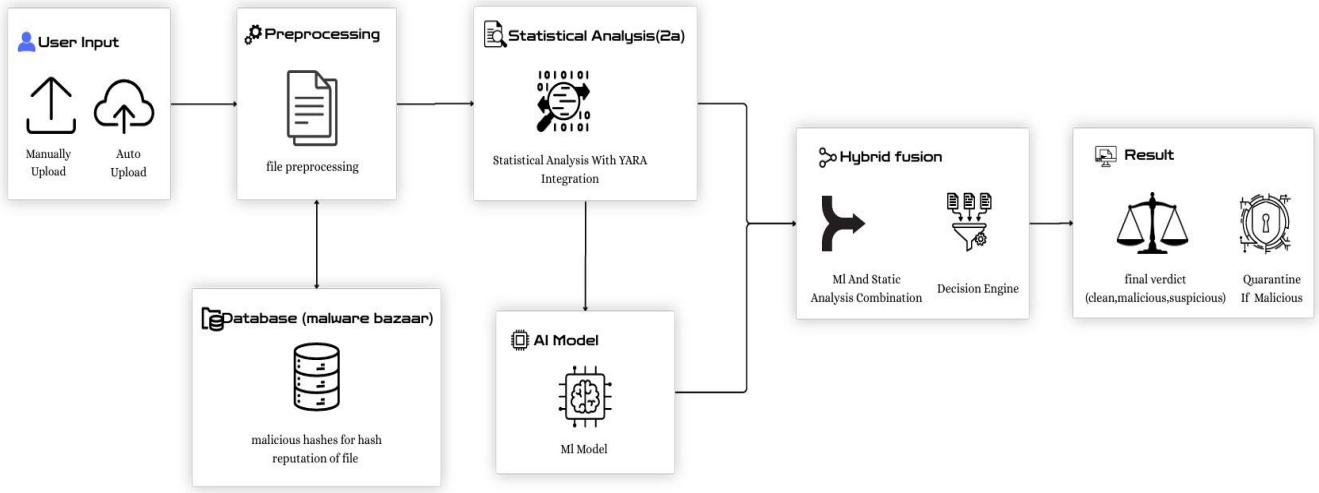

Fig. 1. Pipeline architecture of the proposed SilentEye hybrid static and machine learning framework.

weight from correlated detections across overlay, YARA, entropy, and polyglot layers simultaneously.

Third, a hybrid fusion engine combines outputs from independent heuristic and machine learning analysis streams through a weighted decision engine rather than treating either stream as authoritative in isolation, reducing both false positive and false negative rates relative to single-stream approaches.

B. Preprocessing Layer

The preprocessing layer performs six lightweight validation operations on every submitted file before any deep analysis is initiated. Extension validation checks the file extension against a maintained whitelist covering nine image formats, nine video formats, PDF, and seven document formats, with immediate rejection of executable extensions including .exe, .bat, .js, .vbs, .ps1, and .scr. Double-extension detection flags filenames where a whitelisted extension precedes a blocked executable extension, addressing the common social engineering pattern of files named invoice.pdf.exe or photo.jpg.vbs. Magic byte verification reads the first sixteen bytes of each file and validates the declared format signature against known headers including FF D8 FF for JPEG, 89 PNG for PNG, 25 50 44 46 for PDF, and RIFF for AVI, flagging any mismatch between declared extension and actual file format as suspicious. Unicode normalization strips control characters and bidirectional override sequences from filenames, preventing right-to-left override tricks that visually disguise executable extensions as document names.

Hash reputation lookup computes the SHA-256 digest of each file and queries a locally maintained SQLite database sourced from MalwareBazaar the database is synchronized periodically outside the scanning workflow and therefore does not require internet connectivity during file analysis.

containing over one million confirmed malicious file hashes, as shown in Fig. 1. A hash match produces an immediate malicious verdict bypassing all subsequent analysis stages, providing zero-latency detection for known malware without heuristic computation cost. Files passing all preprocessing checks with no suspicious flags proceed to the statistical analysis engine. Files flagged suspicious by any preprocessing check proceed to both the statistical analysis and machine learning layers for deeper inspection.

C. Statistical Analysis Engine — Stage 2A

The statistical analysis engine implements a two-tier detection architecture comprising a universal layer applicable to all file formats and a format-specific layer providing targeted inspection for each supported file category. The universal layer executes ten detection checks on every file regardless of format. Shannon entropy analysis computes byte-level entropy across sampled file regions using per-category thresholds of 7.98 for image formats and 7.998 for document formats, with JPEG, FLAC, OGG, WAV, and all video formats excluded from entropy analysis entirely due to their naturally high-entropy compressed content. Overlay detection identifies data appended beyond the declared container boundary of each file format. Polyglot detection scans file content for multiple format magic byte signatures, identifying files that simultaneously satisfy the structural requirements of two or more file formats. PE and ELF header detection locates Windows executable and Linux binary signatures

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

within non-executable file content, with is_valid_pe() validation requiring a correct PE offset pointer at position 0x3C to eliminate false positives from statistically inevitable MZ byte sequences in compressed data. Base64 obfuscation detection, XOR-encoded payload detection, and hex-encoded payload detection identify common payload encoding schemes. IOC extraction recovers embedded URLs, IP addresses, and domain strings for reputation analysis against a curated list of known command-and-control infrastructure including ngrok.io, pastebin.com, and duckdns.org. Twenty-nine custom YARA rules organized across eleven categories provide signature-based detection covering executable signatures in media files, polyglot file patterns, suspicious document content, network indicators, and known malware family strings.

Format-specific detection layers extend coverage across all six supported file categories. For image files, four dedicated layers detect executable content appended after JPEG FFD9 and PNG IEND end-of-file markers, PE signatures embedded within EXIF APP marker segments, abnormal PNG canvas dimensions exceeding 8,000 pixels in either dimension, and oversized metadata blobs in JPEG APP markers exceeding 20 KB. For PDF files, five layers parse dangerous action chains combining JavaScript with Launch or OpenAction directives, embedded executable file objects, high-entropy unclassified stream objects, object count anomalies indicative of heap spray patterns, and known CVE-associated exploit signatures including PowerShell invocations, mshta.exe references, and hex-obfuscated JavaScript. For office documents, six layers detect VBA macro markers, OLE binary stream anomalies, external URL relationships, DDEAUTO field injection, exploit delivery command patterns, and LibreOffice StarBasic macro content in ODF formats. For SVG files, four layers detect fifteen dangerous script patterns, XML External Entity injection via DOCTYPE declarations, external resource references pointing to suspicious domains, and base64-encoded executable payloads within data URIs. For video files, four layers validate format-specific container magic bytes, inspect MP4 box structures for malformed size fields, scan metadata atoms for embedded C2 URLs, and detect executable overlays beyond declared RIFF or MP4 container boundaries. For audio files, format-specific overlay detection operates on MP3 sync frame boundaries, WAV RIFF-declared data chunk limits, and FLAC STREAMINFO-computed audio boundaries, with entropy spike detection identifying sudden high-entropy transitions within audio streams indicative of injected payload blocks.

Each detection layer produces a score between 0 and 100 based on signal strength. Artifact deduplication groups correlated layers by the underlying artifact they detect — executable payload, appended payload, high entropy, and network IOC — and retains only the highest-scoring layer within each group before aggregation. Six correlation rules boost scores when independent signals co-occur, including a 60-point boost when metadata anomaly combines with PE detection and a 65-point boost when PDF heap spray combines with suspicious stream detection. Weighted scores map to verdicts through thresholds of 120 and above for malicious, 80 through 119 for suspicious high, 50 through 79 for suspicious medium, and below 30 for clean. Three unambiguous conditions — MalwareBazaar hash match, YARA rule match, and PE detection within file metadata — bypass scoring entirely and produce immediate malicious verdicts.

D. Machine Learning Layer — Stage 2B

The machine learning layer deploys four specialized Random Forest classifiers [12] targeting steganalytic detection across JPEG, PNG, AVI video, and SVG file formats. Random Forest was selected over deep learning architectures for four operationally motivated reasons: interpretability through feature importance scores enabling explainable detection decisions, training and inference on CPU without GPU requirement, sub-200 millisecond single-file inference latency, and resistance to overfitting through bagging and feature subsampling [12].

Each classifier operates on a format-specific feature vector of thirty hand-crafted domain features rather than raw pixel or byte values. The JPEG classifier extracts five LSB chi-square features [1], five pixel statistics, five alpha channel statistics, five DCT quantization table features targeting the low-value QT signature produced by steghide re-encoding at quality 95 through 97, two RS analysis features [2], two inter-channel correlation features, one block uniformity feature, and five JUNIWARD DCT coefficient histogram features. The PNG classifier uses the same thirty-feature vector with primary detection weight on alpha channel features, specifically alpha_midrange_ratio which measures the fraction of pixels with transparency values between 0 and 255 exclusive, a near-binary signal separating legitimate transparent images from alpha channel stego content. The video classifier extracts twenty features from container structure, entropy distribution across five sampled 64 KB regions, embedded URL patterns, executable header detection, and overlay size measurement without requiring video decoding. The SVG classifier extracts twenty features covering script and JavaScript execution patterns, base64 encoding anomalies, structural complexity indicators, and external resource references.

All four classifiers use identical hyperparameters of 100 estimators, maximum depth 6, minimum samples per leaf 2, square root feature subsampling, and balanced class weights with random state 42 for full reproducibility.

E. Hybrid Fusion Engine

The hybrid fusion engine receives scored outputs from both the statistical analysis engine and the machine learning layer and produces a single unified verdict as shown in Fig. 1. The fusion logic implements a two-case decision table. In Case A, where the statistical analysis engine reports zero detections, the machine learning verdict is treated as final. In Case B, where the statistical analysis engine reports one or more detections, the decision engine combines both streams through weighted aggregation with the heuristic score carrying primary weight and the ML probability score contributing a supplementary signal. This asymmetric weighting reflects the higher specificity of rule-based detections against known attack patterns relative to the probabilistic nature of ML classification on novel inputs.

Files receiving a final malicious verdict are automatically quarantined. Files receiving a suspicious verdict are flagged for user review with detailed per-layer detection reasoning. Files receiving a clean verdict are released without restriction. All scan results including per-layer scores, detected artifacts, and final verdicts are logged locally in SQLite without transmission to external servers, preserving the offline-first privacy model of the system.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

IV. METHODOLOGY

A. Evaluation Strategy

SilentEye was evaluated through two independent experimental tracks corresponding to its two detection engines. The statistical analysis engine, Stage 2A, was evaluated on a self-constructed dataset of clean and malicious files designed to measure true positive rate, false positive rate, and scan latency across all six supported file categories under realistic attack conditions. The machine learning layer, Stage 2B, was evaluated separately on established academic steganalysis benchmarks to enable direct comparability with published steganalysis literature. Both tracks used disjoint train and test partitions with fixed random seeds to ensure reproducibility.

B. Stage 2A Dataset Construction

The clean dataset comprises 1,805 files sourced from established academic and open-source repositories across six categories. Images were drawn from COCO, BOSSBase, DIV2K, and ALASKA2 [6], [7], [9], supplemented with 143 live images captured on mobile devices and received through WhatsApp to capture real-world compression artifacts specific to the Indian mobile messaging context. Audio files were sourced from the GTZAN genre classification dataset covering ten music genres and UrbanSound8K for urban sound recordings. Video files were drawn from UCF101 [8] using MJPEG-encoded AVI files and a Kaggle video dataset including 4K UHD H.264 content. Document files were sourced from enterprise DOCX samples. PDF files were drawn from Cornell University arXiv preprint papers, selected specifically to represent complex real-world academic documents with high object counts that stress-test false positive resilience. SVG files were sourced from the SVGRepo open-source collection. An additional twenty-five benign edge-case files were constructed to probe false positive resilience, including PDF forms with legitimate field-validation JavaScript, academic PDFs with 1,500 to 2,500 objects, DOCX files with disabled macro comments, and JPEG files with large legitimate EXIF blocks from DSLR cameras.

The malicious dataset comprises 1,780 files generated through controlled injection of attack patterns into the clean dataset at approximately a 1:1 ratio, implementing twenty-nine distinct attack types across the six file categories. Image attacks include polyglot PE overlay, entropy overlay injection, EXIF PE injection, oversized metadata blobs, embedded IOC URLs, and abnormal PNG dimensions. Audio attacks include FLAC PE overlay, WAV end-of-file overlay, WAV PE overlay, and WAV LSB steganography. Video attacks include AVI RIFF overlay, MP4 PE overlay, corrupted container magic bytes, and malformed MP4 box structures. Document attacks include VBA macro injection, external relationship exploitation, OLE macro embedding, XML anomaly injection, and exploit signature patterns. PDF attacks include JavaScript-Launch chains, OpenAction exploitation, heap spray patterns, high-entropy stream injection, and embedded executable objects. SVG attacks include plain and obfuscated script injection, XXE entity injection, and external PE payload references.

It should be noted that the malicious dataset primarily consists of controlled attack-pattern injections designed to evaluate detector coverage under reproducible conditions. While this methodology enables systematic testing across diverse attack classes, it should not be interpreted as representative of large-scale operational malware campaigns. Evaluation against real-world malware collections remains future work.

TABLE I:
STAGE 2A STATIC ANALYSIS DATASET COMPOSITION

Category	Clean Files	Malicious Files	Total	Source
Image	500	500	1,000	COCO, BOSSBase, DIV2K, ALASKA2, 143 live WhatsApp images
Audio	300	300	600	GTZAN (150 WAV), UrbanSound8K (150 FLAC)
Video	300	300	600	UCF101 (150 AVI), Kaggle dataset (150 MP4, 4K UHD)
Document	180	180	360	Enterprise DOCX corpus
PDF	200	200	400	Cornell University arXiv preprints
SVG	300	300	600	SVGRepo open-source collection
Edge cases	25	—	25	Synthetic benign worst-case constructs
Total	**1,805**	**1,780**	**3,585**	—

C. Stage 2B Dataset Construction

Two datasets were used for machine learning evaluation. The development dataset, used for initial feature validation and architecture design, comprises 2,666 files spanning JPEG steghide samples, PNG alpha channel samples, AVI video samples, and SVG samples at varying clean-to-malicious ratios. The final research dataset, used for all reported paper results, comprises 36,800 files drawn from six established benchmark sources. ALASKA2 [6] contributed 20,000 JPEG files embedded using DCT and JUNIWARD adaptive steganography [4]. BOSSBase [7] contributed 8,400 PNG files using alpha channel LSB embedding. DIV2K [9] contributed 1,600 PNG files using the same embedding method. A flat-field calibration dataset contributed 200 TIFF files. UCF101 [8] contributed 4,000 AVI files using RIFF chunk and overlay injection. A custom-constructed SVG dataset contributed 2,600 files using JavaScript, XXE, and PE injection techniques.

All datasets used an 80/20 stratified train-test split with random_state=42, ensuring train and test partitions contain completely disjoint files with no data leakage. Table I summarizes the per-dataset split.

TABLE II:
TRAIN/TEST SPLIT PER DATASET

Dataset	Format	Train Clean	Train Malicious	Test Clean	Test Malicious	Total
ALASKA2	JPEG	8,000	8,000	2,000	2,000	20,000
BOSSBase	PNG	3,360	3,360	840	840	8,400
DIV2K	PNG	640	640	160	160	1,600

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

Dataset	Format	Train Clean	Train Malicious	Test Clean	Test Malicious	Total
Flat-field	TIFF	80	80	20	20	200
UCF101	AVI	1,600	1,600	400	400	4,000
SVG Custom	SVG	1,040	1,040	260	260	2,600
Total	—	**14,720**	**14,720**	**3,680**	**3,680**	**36,800**

Two evaluation scenarios were considered for JPEG steganalysis. The first used a development dataset containing Steghide-generated JPEG samples to validate feature effectiveness under controlled conditions. The second used the ALASKA2 benchmark containing J-UNIWARD adaptive steganography, representing a significantly more challenging and realistic evaluation scenario.

D. Feature Extraction

Each file format uses a fixed thirty-feature vector for image formats and twenty-feature vectors for video and SVG formats, computed once per file and shared across both the statistical analysis and machine learning engines to eliminate redundant I/O and computation. Feature groups for image formats include LSB chi-square statistics [1], global pixel distribution statistics, alpha channel statistics, DCT quantization table statistics targeting steghide's high-quality re-encoding signature, RS analysis estimators [2], inter-channel correlation coefficients, JPEG block uniformity measures, and JUNIWARD-specific DCT coefficient histogram features [4]. Video features are extracted from container structure, sampled entropy regions, and tail-region byte patterns without requiring video decoding, enabling format-agnostic analysis without FFmpeg dependency. SVG features are extracted from script pattern density, encoding anomaly indicators, structural complexity metrics, and external resource reference counts.

E. Mathematical Formulation

1) Entropy Analysis: Shannon entropy is computed over each sampled byte region as

$$H(X) = -\sum_{i=0}^{255} p(x_i)\log_2 p(x_i) \quad (1)$$

where $p(x_i)$ denotes the observed frequency of byte value x_i within the sampled region. A region is flagged as anomalous when $H(X)$ exceeds the category-specific threshold τ_c, with $\tau_{image} = 7.98$ and $\tau_{doc} = 7.998$ bits per byte. JPEG, FLAC, OGG, WAV, and all video formats are excluded from entropy analysis due to their naturally high-entropy compressed content.

2) Weighted Risk Score with Artifact Deduplication: Each heuristic detection layer l produces a binary detection indicator $d_l(f) \in \{0,1\}$ with an associated weight $w_l \in [0,100]$. A single underlying artifact frequently triggers multiple correlated layers simultaneously — for instance, an embedded PE header inside a JPEG file is independently flagged by pe_detection, yara_detection, base64_obfuscation, and overlay_detection. Summing these contributions directly would inflate the score in proportion to the number of layers triggered rather than the severity of the artifact itself. To prevent this, layers are partitioned into artifact-correlated groups $g \in G$, and only the highest-weighted positive detection within each group contributes to the aggregate score:

$$S(f) = \sum_{g \in G} \max_{l \in g} (w_l \cdot d_l(f)) \quad (2)$$

Under Eq. (2), the executable-payload group in the example above contributes only its maximum value of 80, rather than the uncorrected sum of 280 across all four layers. A correlation boost term $C(f)$ is then added when independent signal pairs co-occur, with fixed boost values determined empirically: entropy combined with polyglot detection (+20), Base64 combined with PE detection (+50), network IOC combined with domain detection (+25), metadata anomaly combined with PE detection (+60), macro combined with external relationship (+55), and PDF heap spray combined with suspicious stream entropy (+65). The total weighted score is therefore $S_{total}(f) = S(f) + C(f)$.

Equation (2) solves an aggregation problem: it converts a set of independent binary signals into a single deduplicated score. This score must then be converted into a categorical verdict, which is a distinct operation requiring a separate mapping rather than a continuation of the same arithmetic. The verdict from the statistical analysis engine is therefore determined by

$$V(f) = \begin{cases} \text{malicious} & S_{total}(f) \geq 120 \\ \text{suspicious}_{high} & 80 \leq S_{total}(f) < 120 \\ \text{suspicious}_{med} & 50 \leq S_{total}(f) < 80 \\ \text{suspicious}_{low} & 30 \leq S_{total}(f) < 50 \\ \text{clean} & S_{total}(f) < 30 \end{cases} \quad (3)$$

Three unambiguous conditions bypass Eq. (3) and yield an immediate malicious verdict regardless of $S_{total}(f)$: a SHA-256 hash match against the MalwareBazaar reputation database, a positive YARA rule match, and confirmed PE detection within file metadata.

The internal score $S_{total}(f)$ is unbounded above 100 by construction, since correlation boosts are additive on top of an already-maximal group score. The verdict tier $V(f)$ from Eq. (3), rather than the raw score itself, is what determines the bounded display value $F(f) \in [0,100]$ reported to the user interface:

$$F(f) = \begin{cases} 85 & \text{if } V(f) = \text{malicious} \\ 65 & \text{if } V(f) = \text{suspicious}_{high} \\ 45 & \text{if } V(f) = \text{suspicious}_{med} \\ 30 & \text{if } V(f) = \text{suspicious}_{low} \\ 0 & \text{if } V(f) = \text{clean} \end{cases} \quad (3a)$$

This decoupling between the unbounded internal accumulation score and the bounded UI display score ensures that scores exceeding 100 internally, which arise naturally from correlation boosting, never appear as out-of-range values in the user-facing verdict.

3) Hybrid Fusion Decision Rule: Equations (2) and (3) resolve disagreement *within* the heuristic engine by aggregating its internal signals into a single verdict. A separate problem arises *between* engines: the heuristic verdict $V(f)$ and the machine learning verdict $V_{ml}(f)$, produced independently by the corresponding format-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

specific Random Forest classifier, may agree or conflict, and this conflict cannot be resolved by the same scoring arithmetic used in Eq. (2), since $V_{ml}(f)$ is not expressed on the same additive point scale as $S_{total}(f)$. The hybrid fusion engine therefore applies a distinct decision rule:

$$V_{final}(f) = \begin{cases} V_{ml}(f) & \text{if } S_{total}(f) = 0 \text{(Case A)} \\ \text{Decision Table}(V(f), V_{ml}(f)) & \text{if } S_{total}(f) > 0 \text{(Case B)} \end{cases} \quad (4)$$

In Case A, the heuristic engine reports no detections, leaving no conflict to resolve, and the machine learning verdict is accepted as final. In Case B, the heuristic engine reports at least one detection, and the final verdict is resolved through a discrete decision table that jointly considers the heuristic verdict tier from Eq. (3) and the machine learning verdict, escalating the final classification when both engines agree and applying engine-priority rules when they diverge. This is formalized as a decision rule rather than a closed-form arithmetic expression because the two verdicts being combined are categorical outputs from heterogeneous detection mechanisms, not commensurate numerical quantities that could be meaningfully averaged or weighted-summed.

4) Evaluation Metrics: Detection performance reported in Section VI follows standard classification metric definitions:

$$\text{Precision} = \frac{TP}{TP + FP}, \text{Recall} = \frac{TP}{TP + FN} \quad (5)$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \text{FPR} = \frac{FP}{FP + TN} \quad (6)$$

where TP, FP, FN, and TN denote true positive, false positive, false negative, and true negative counts respectively.

F. Model Training Configuration

All four Random Forest classifiers [12] were trained using scikit-learn with 100 estimators, maximum tree depth of 6, minimum samples per leaf of 2, square root feature subsampling at each split, and balanced class weighting to address any residual class imbalance after stratified splitting. Maximum depth was constrained to 6 to balance model capacity against overfitting risk, permitting at most 64 leaf nodes across the 30-dimensional feature space — sufficient to capture feature interactions without memorizing training noise. Five-fold cross-validation was performed on the training partition for each model to assess result stability independent of the held-out test set.

G. Reproducibility

All experiments use fixed random seeds, specifically random_state=42 for scikit-learn model training and np.random.default_rng(42) for dataset shuffling operations, ensuring deterministic and fully reproducible results given identical input datasets. Dataset paths are resolved through an environment variable configuration rather than hardcoded paths, supporting evaluation across heterogeneous deployment environments.

V. RESULTS

A. Stage 2A Heuristic Engine Performance

Table III reports detection performance of the statistical analysis engine on the malicious dataset described in Table II. The engine achieved an overall true positive rate of 93.8% across 1,780 malicious files at an average scan latency of 166 milliseconds.

TABLE III:
STAGE 2A DETECTION PERFORMANCE--MALICIOUS DATASET

Category	Files	Detected	TP Rate	Avg Scan Time
SVG	300	300	100.0%	13.0 ms
Video	300	300	100.0%	41.2 ms
DOCX	180	180	100.0%	51.1 ms
PDF	200	199	99.5%	1,276 ms
Images	500	452	90.4%	17.3 ms
Audio	300	238	79.3%	18.0 ms
Overall	**1,780**	**1,669**	**93.8%**	**166 ms**

Three categories — SVG, video, and DOCX — achieved 100% detection due to the structurally explicit nature of the injected attack patterns in these formats. PDF achieved near-perfect detection at the cost of the highest scan latency, a direct consequence of the full-file read requirement for stream pattern matching described in Section III-C. Audio recorded the lowest detection rate, attributable to the disabled LSB detection layer discussed in Section VII-A.

Table IV reports false positive performance on the clean dataset.

TABLE IV:
STAGE 2A FALSE POSITIVE PERFORMANCE--CLEAN DATASET

Category	Files	FP Count	FP Rate	Avg Scan Time
SVG	300	0	0.0%	8.1 ms
Audio	300	0	0.0%	14.0 ms
Video	300	0	0.0%	41.3 ms
DOCX	180	1	0.6%	43.2 ms
Images	500	5	1.0%	12.9 ms
PDF	200	5	2.5%	1,610 ms
Edge cases	25	4	16.0%	13.2 ms
Overall	**1,805**	**15**	**0.8%**	**197 ms**

The overall false positive rate of 0.8% across 1,805 clean files satisfies typical production deployment thresholds for static security scanners. The elevated 16.0% false positive rate on the edge-case

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

subset is isolated to twenty-five deliberately constructed worst-case inputs and is analyzed separately in Section VI-D; it does not reflect performance on the representative clean corpus, where the rate is 0.6%.

B. Resource Efficiency

Table V reports memory consumption and scan latency under the smart-read architecture described in Section III-B.

TABLE V:
RESOURCE EFFICIENCY UNDER SMART-READ ARCHITECTURE

File	Actual Size	RAM Used	Scan Time
4K UHD MP4 video	~800 MB	~3.5 MB	<200 ms
GTZAN WAV audio	~2.6 MB	91 KB	<20 ms
arXiv PDF	~5 MB	5 MB (full)	~1,300 ms
DOCX document	~200 KB	200 KB (full)	~50 ms
JPEG image	~3 MB	68 KB	<20 ms

An 800 MB video file is scanned using approximately 3.5 MB of RAM, a compression ratio exceeding 200:1 between file size and memory footprint. PDF and DOCX formats require full-file reads due to stream and XML parsing requirements respectively, explaining their proportionally higher memory usage relative to media formats that benefit from header-plus-tail sampling.

C. Stage 2B Machine Learning Performance

Table VI reports test set performance of the four Random Forest classifiers on the final research dataset of 36,800 files described in Table I.

TABLE VI:
STAGE 2B MACHINE LEARNING PERFORMANCE — FINAL DATASET

Model	Train F1	CV F1	Test F1	FPR	FNR	Test Files
image_jpg	0.6525	0.4943	0.5231	60.65%	42.84%	3,977
image_png_alpha	1.0000	1.0000	**1.0000**	0.00%	0.00%	2,037
video	1.0000	1.0000	**1.0000**	0.00%	0.00%	800
svg	1.0000	1.0000	**1.0000**	0.00%	0.00%	520

Dataset	Embedding Method	F1
Development Dataset	Steghide	0.9965
ALASKA2	J-UNIWARD	0.5231

The substantial performance gap between Steghide and J-UNIWARD demonstrates that the proposed Random Forest classifier primarily benefits from quantization-table artifacts introduced by JPEG recompression. Adaptive algorithms such as J-UNIWARD minimize these artifacts and therefore remain difficult to detect using handcrafted statistical features.

Three of four classifiers achieved perfect test F1 scores. This result requires explicit justification rather than acceptance at face value, since perfect classification scores are a recognized indicator of potential data leakage or trivial class separability in machine learning evaluation.

For PNG alpha channel detection, the alpha_midrange_ratio feature produces a near-binary signal: legitimate transparent images exhibit alpha values clustered at 0 or 255, while stego-embedded images exhibit a midrange ratio between 0.25 and 0.75 depending on payload size. This is a deterministic structural property of the embedding method rather than a learned statistical artifact. For AVI video detection, the overlay_size_norm feature is effectively binary, with clean UCF101 videos exhibiting zero overlay and malicious files exhibiting nonzero overlay by construction of the injection method. For SVG detection, the danger_pattern_density and js_exec_density features are near-zero for clean files and strongly positive for malicious files due to the explicit presence of script tags, event handlers, and XXE entities absent from clean SVG sources. All three results were confirmed stable under five-fold cross-validation with CV F1 = 1.0 ± 0.0, and all datasets used strict 80/20 splits with disjoint files as reported in Table I, eliminating data leakage as an explanation.

The JPEG classifier presents a markedly different result. Test F1 of 0.5231 with false positive and false negative rates exceeding 40% indicates performance close to random chance on the ALASKA2 dataset, which includes J-UNIWARD adaptive steganography [4]. This is consistent with the development dataset result in Table VII, where the same architecture achieved F1 = 0.9965 on steghide-only samples, confirming that detection capability is present for tool-based steganography but absent for adaptive embedding.

TABLE VII:
STAGE 2B DEVELOPMENT DATASET RESULTS (STEGHIDE-ONLY)

Model	F1 Score	Precision	Recall	FPR	FNR
image_jpg (steghide)	**0.9965**	1.0000	0.9931	0.00%	0.69%
image_png_alpha	**1.0000**	1.0000	1.0000	0.00%	0.00%
video	**0.9895**	0.9787	0.9895	0.69%	1.05%
svg	**1.0000**	1.0000	1.0000	0.00%	0.00%

D. Feature Importance

Table VIII reports the top feature importance scores for the two image-based classifiers, confirming that detection decisions are driven by the theoretically motivated features described in Section III-D rather than spurious correlations.

TABLE VIII:
TOP FEATURE IMPORTANCE — IMAGE CLASSIFIERS

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

image_jpg Feature	Importance	image_png_alpha Feature	Importance
dct_qt_std	0.2556	alpha_mean	0.2747
dct_chroma_diff	0.2472	alpha_std	0.2117
dct_qt_sum	0.2286	alpha_present	0.1961
dct_qt_mean	0.1701	alpha_midrange_ratio	0.1835
dct_qt_maxmin	0.0685	alpha_binary_ratio	0.1308

All five top features for image_jpg belong to the DCT quantization table feature group, confirming that the steghide-detection capability of this classifier is driven entirely by quantization table analysis rather than pixel-domain statistics. Correspondingly, all top features for image_png_alpha are alpha channel statistics, confirming that detection is structurally grounded rather than incidental.

VI. LIMITATIONS AND FUTURE WORK

A. Limitations

Audio LSB detection remains disabled due to a 55.3% false positive rate on the GTZAN music dataset; chi-square analysis [1] cannot separate steganographic embedding from the naturally uniform LSB distributions in complex music, consistent with [3]. This is reflected in Table III's 79.3% audio detection rate, the lowest among all categories.

The JPEG classifier achieves only F1 = 0.5231 on J-UNIWARD samples [4], against F1 = 0.9965 on steghide alone (Table VII). This matches published CNN benchmarks — SRNet at 64.3% [5] and EfficientNet-B4 at 68.1% [13] — confirming hand-crafted features cannot capture adaptive embedding without GPU-based deep learning.

The strong performance observed on Steghide-generated samples (F1 = 0.9965) should therefore be interpreted as detection of tool-specific recompression artifacts rather than a generalized capability against adaptive steganographic algorithms.

A 16.0% false positive rate on a synthetic edge-case subset traces to four DOCX files where YARA correctly flagged VBA binary markers in macros that were disabled but not removed, a distinction current static parsing cannot make. This does not reflect the 0.6% rate on the representative corpus in Table IV.

SilentEye performs static analysis only; runtime behaviors such as macro-triggered network calls or process injection are not monitored. Real-time scanning is also unsupported on Android 14 and iOS due to OS background-access restrictions, limiting protection on the mobile platforms where WhatsApp and Instagram threats are most prevalent.

The img_bad_dimensions attack achieved only 2.4% detection, traced to a dataset construction issue — most variant files were JPEG, where pixel dimension checking does not apply — rather than a layer failure, which was confirmed functional on pure PNG inputs separately.

PDF scanning requires full-file reads, producing average scan times of 1,276–1,610 ms, an order of magnitude slower than any other format (Tables III, IV). Finally, all four classifiers are trained on specific tool signatures; detection of unrepresented steganographic tools cannot be guaranteed without retraining.

B. Future Work

Closing the JUNIWARD gap requires integrating a CNN-based model such as SRNet [5] or EfficientNet [13] as an optional GPU-accelerated module for low-confidence cases. The audio limitation calls for genre-aware LSB baselines or spectrogram-based CNN analysis [3]. The DOCX edge case requires VBA project binary parsing to distinguish disabled from active macros. Behavioral sandbox integration would address fileless and runtime threats outside current static scope. Mobile constraints point toward an Android-native foreground service or direct messaging-API interception rather than folder polling. PDF latency can be reduced through selective stream sampling targeting anomalous objects instead of exhaustive parsing. Finally, the existing MalwareBazaar hash pipeline (Section III-B) provides a natural mechanism for surfacing and incorporating novel steganographic tool samples through periodic retraining.

VII. CONCLUSION

This paper presented SilentEye, an offline multi-stage detection framework for malware concealed in multimedia files shared through WhatsApp and Instagram. The system combines a YARA-driven heuristic engine across six file categories with four Random Forest classifiers, fused through a hybrid decision engine into clean, suspicious, or malicious verdicts.

The heuristic engine achieved 93.8% true positive rate and 0.8% false positive rate across 3,585 files at 166 ms average latency, scanning 800 MB videos within 3.5 MB RAM. The ML layer achieved F1 = 0.9965 for JPEG steghide and F1 = 1.0000 for PNG, video, and SVG detection across 36,800 benchmark files [6], [7], with perfect scores structurally justified rather than assumed.

The main limitation — failure on J-UNIWARD adaptive steganography [4] — matches published CNN benchmarks [5], [13] and is addressed in defined future work rather than left unacknowledged. SilentEye is therefore scoped as a static pre-screening layer for tool-based steganography and container attacks, not a replacement for behavioral or GPU-based adaptive steganalysis.

By unifying heuristic and ML detection across six formats in a single offline pipeline, SilentEye shows that practical protection against media-borne malware is achievable without cloud dependency or specialized hardware — directly relevant to the air-gapped and mobile-first threat context that motivated this work.

SilentEye should be viewed as a practical static pre-screening framework for media-borne threats rather than a replacement for behavioral malware analysis or advanced GPU-based adaptive steganalysis systems. The framework demonstrates strong performance against tool-based steganography, container abuse, embedded executables, and document-based attacks, while adaptive algorithms such as J-UNIWARD remain an open challenge.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

VIII. REFERENCES

[1] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," in Proc. 3rd Int. Workshop Information Hiding, vol. 1768, Springer,1999,pp.61–76.[Online].Available: https://ieeexplore.ieee.org/document/648030

[2] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," in Proc. ACM Workshop Multimedia and Security, Ottawa, Canada, 2001, pp. 27–30. [Online]. Available: https://dl.acm.org/doi/10.1145/1232454.1232466

[3] A. D. Ker, "A general framework for the statistical steganalysis of LSB embedding," in Proc. 9th Int. Workshop Information Hiding, vol. 4567, Springer, 2007, pp. 296–311. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-77370-2_20

[4] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," EURASIP Journal on Information Security, vol. 2014, no. 1, pp. 1–13, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6595569

[5] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," IEEE Transactions on Information Forensics and Security, vol. 14, no. 5, pp. 1181–1193, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8470101

[6] R. Cogranne, Q. Giboulot, and P. Bas, "The ALASKA steganalysis challenge: A first step towards steganalysis into the wild," in Proc. ACM Workshop on Information Hiding and Multimedia Security, Paris, France, 2019, pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/3335203.3335238

[7] P. Bas, T. Filler, and T. Pevný, "'Break our steganographic system' — the ins and outs of organizing BOSS," in Proc. 13th Int. Workshop Information Hiding, vol. 6958, Springer, 2011, pp. 59–70. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-24178-9_5

[8] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," arXiv preprint arXiv:1212.0402, 2012. [Online]. Available: https://arxiv.org/abs/1212.0402

[9] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in Proc. IEEE CVPR Workshops, Honolulu, HI, USA, 2017, pp. 126–135. [Online]. Available: https://ieeexplore.ieee.org/document/8014884

[10] R. Chaganti, V. Ravi, and T. D. Pham, "A deep learning based approach for detecting novel steganography techniques in digital images," arXiv preprint arXiv:2104.05480, 2021. [Online]. Available: https://arxiv.org/abs/2104.05480

[11] F. Strachanski, N. Strodthoff, and S. Schneider, "A comprehensive pattern-based overview of stegomalware," in Proc. ARES CUING Workshop, Vienna, Austria, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3664476.3670439

[12] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324

[13] Y. Yousfi, J. Butora, J. Fridrich, and P. Bas, "Breaking ALASKA: Color images steganalysis using a pre-trained CNN feature extractor," in Proc. IS&T Electronic Imaging, Digital Forensics and Watermarking, 2020, pp. 413-1–413-7. [Online]. Available: https://doi.org/10.2352/ISSN.2470-1173.2020.4.DFWM-413

[14] A. Selvaraj, A. Ezhilarasan, S. L. J. Wellington, and A. Roy Sam, "Digital image steganalysis: A survey on paradigm shift from machine learning to deep learning based techniques," IET Image Processing, vol. 14, no. 12, pp. 2393–2411, 2020. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-ipr.2020.0543

[15] G. Xu, H.-Z. Wu, and Y.-Q. Shi, "Structural design of convolutional neural networks for steganalysis," IEEE Signal Processing Letters, vol. 23, no. 5, pp. 708–712, May 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7444146

[16] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," IEEE Transactions on Information Forensics and Security, vol. 12, no. 11, pp. 2545–2557, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7937836

[17] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep learning for steganalysis via convolutional neural networks," in Proc. SPIE Media Watermarking, Security, and Forensics, vol. 9409, San Francisco, CA, USA, 2015. [Online]. Available: https://doi.org/10.1117/12.2083479

[18] M. Yedroudj, F. Comby, and M. Chaumont, "Yedroudj-Net: An efficient CNN for spatial steganalysis," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Calgary, Canada, 2018, pp. 2092–2096. [Online]. Available: https://ieeexplore.ieee.org/document/8461438

[19] O. Yudin, Y. Symonychenko, and A. Symonychenko, "The method of detection of hidden information in a digital image using steganographic methods of analysis," in Proc. IEEE Int. Conf. Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 2019, pp. 108–112. [Online]. Available: https://ieeexplore.ieee.org/document/9030454

[20] S. Lyu and H. Farid, "Steganalysis using higher-order image statistics," IEEE Transactions on Information Forensics and Security, vol. 1, no. 1, pp. 111–119, Mar. 2006. [Online]. Available: https://ieeexplore.ieee.org/document/1597126