International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(3s) 2026 pp. 142-161
https://doi.org/10.70917/ijcisim-2026-2285
Article

Performance Evaluation And Deep Learning Steganalysis Of Covert APK Transmission Using Video Steganography Using 2D Haar Discrete Wavelet Transform And LSB Substitution

Nilima Dongre[1], Ponmalar G[2], Pavan Prajapati[3], Nityam Padwal[4], Niraj Karande[5]

Ramrao Adik Institute of Technology, D Y Patil deemed to be University[1], Ramdeobaba university[2], Ramrao Adik Institute of Technology,D Y Patil deemed to be University[13,4,5]
*Corresponding author: neilima.dongre@gmail.com[1] [Orcid ID: 0000-0002-0141-290X], ponmalarme@gmail.com[2], pavanprajapati10148@gmail.com[3]; nityampadwal933@gmail.com [4]; nirajkarande137@gmail.com [5]

Abstract: The hidden payload recovery should be exact rather than approximated as there is zero tolerance for byte corruption in APK files. WE aim to develop a system which ensure zero-byte-error recovery for the APK while keeping the stego-video visually indistinguishable from a normal video. We have developed a complete pipeline for covert Android Package file transmission through video steganography, addressing the one engineering requirement that distinguishes APK embedding from every prior payload class in the field: Android's PackageManager rejects any recovered binary containing a single incorrect byte, making zero-error recovery a hard constraint rather than a quality target. The research demonstrates that Hamming (7,4) forward error correction at code rate 4/7 successfully satisfies this constraint across the embedding-extraction cycle, protecting the APK byte stream before LH sub-band coefficient modification and reconstructing it without residual errors under lossless storage conditions. The implementation of 2D Haar Discrete Wavelet Transform embedding in the horizontal-detail LH sub-band proved effective at maintaining imperceptibility despite the elevated payload densities that APK file sizes require, achieving PSNR of 47.3 to 52.1 dB, SSIM above 0.99, NPCR of 89.66%, and UACI of 0.82% within the accepted imperceptibility range. The implementation of VGG16 fine-tuning for spatial LH artefact detection proved effective at classifying stego frames with 91.52% accuracy and F1-score of 91.42%, successfully distinguishing embedded from clean frames despite the absence of prior training data for this specific payload class. The results have validated the effectiveness of frequency-domain binary payload embedding and highlight the potential for neural steganalysis as a practical detection mechanism for APK-in-video covert channels. The integration of all three pipelines into a single deployable desktop GUI application represents an advance not present in any prior covert APK transmission system.

Keywords: video steganography, APK binary embedding, 2D Haar DWT, LSB substitution, Hamming (7,4) error correction, XOR encryption, Steganalysis, VGG16 fine-tuning, CNN-LSTM, NPCR, UACI, Shannon entropy, Android security.

1. Introduction

We have built this system in response to a specific gap in published video steganography research: every documented system treats payload recovery as a quality problem, where a small number of bit errors degrades the result gracefully and the receiver still obtains something usable. This assumption holds for text, image, and audio payloads. It fails categorically for Android Package files. An APK binary carries four independent validation layers — a DEX bytecode SHA-1 checksum, a binary-encoded AndroidManifest.xml with chunk-header constraints, ARSC resource tables with alignment requirements, and a ZIP central directory with offset records — and Android's PackageManager verifies all four before permitting installation. One wrong byte in any layer produces a hard parse

rejection with no degraded fallback. The research demonstrates that this correctness requirement demands a fundamentally different pipeline design from the ones documented in the video steganography literature [1,3,17,18].

The field has recognised related problems. Error-correcting codes appear in DCT-domain systems [9] to resist codec compression noise, and covert APK channel research [2,10,11,26] has established that Android binaries can traverse standard inspection tools undetected. The implementation of these two lines of work in the same system has not been reported, however, and neither body of research has treated byte-level binary correctness as the organising design constraint. We have filled that gap by combining Hamming (7,4) algebraic error correction with 2D Haar DWT LH-band embedding, applying XOR scrambling across the encoded stream, and training two neural detectors — a fine-tuned VGG16 spatial classifier and a CNN-LSTM temporal model — against the resulting stego signal.

The results have validated the effectiveness of this approach across three evaluation dimensions. The embedding pipeline maintains perceptual quality consistently above published imperceptibility thresholds despite APK payload densities. The VGG16 detector achieves competitive accuracy against a novel payload class for which no prior training benchmark exists. The CNN-LSTM hybrid reveals a fundamental tension between scrambling-based security and temporal detection that the field has not previously documented for binary payload steganography. The complete system, including embedding, extraction, and steganalysis, is packaged as a desktop application accessible without command-line interaction.

Section 2 reviews the literatures that informed specific design decisions. Section 3 documents the system's known limitations with their causes. Sections 4 and 5 describe the pipeline and its algebraic foundations. Section 6 defines the evaluation metrics. Sections 7 and 8 present measured results and compare them against eleven verified prior systems. Section 9 identifies the four experiments that would resolve the principal remaining unknowns.

2. Literature Survey

We have reviewed four adjacent literatures to ground the design decisions in this system: survey-level taxonomies of video steganography, DWT-based frequency-domain embedding, algebraic error correction for covert channels, and prior covert APK transmission work. A 2024 search of IEEE Xplore, Springer Link, arXiv, and ACM Digital Library found no published work at the intersection of all four. The research demonstrates that each adjacent literature contributes specific, actionable knowledge while stopping short of the full problem this system addresses.

2.1 Survey taxonomies: what they provided and where they stopped

We have drawn on the major video steganography surveys [1,3,17,18] primarily for their documentation of the capacity-imperceptibility-robustness trilemma, which organises the field's design trade-offs and provided the vocabulary for stating our own trade-off explicitly: this system prioritises imperceptibility and binary payload integrity over compression robustness. The surveys' consistent focus on degradable payload types meant that byte-level correctness was absent as a design axis, and the research demonstrates that adding it changes several design decisions that the trilemma framework alone would not surface. The H.264-stage analysis from Mstafa and Elleithy [17,18] proved directly useful: their finding that luminance modifications at low amplitude produce less perceptual distortion than equal-energy chrominance modifications determined the YCbCr separation and Y-channel-only embedding in the system described in Section 4.

2.2 DWT embedding systems: design decisions the comparisons resolved

We have compared our sub-band and frame-distribution choices against the two most relevant prior DWT systems and found that each comparison resolved a specific design decision. Ernawan's adaptive system [21] implements scene-change frame selection, achieving improved per-frame PSNR at the cost of reduced total capacity. The implementation of uniform frame distribution in our system, rather than scene-change selection, proved necessary for APK file sizes that routinely exceed the capacity available in scene-change frames alone; the trade-off accepts higher total modification in exchange for predictable capacity scaling. The sub-band comparison between Ernawan's LL choice and our LH choice reflects the energy-distortion relationship between the two bands: LL modifications at APK payload densities produce visible distortion, while LH modifications propagate through inverse Haar synthesis to pixel-domain changes below one intensity unit.

The research demonstrates that further refinements validated by Pilania et al. [22,23] and Alobaidi and Mikhael [24] — region-of-interest coefficient restriction and adaptive within-band weighting, respectively — represent proven improvements applicable to future versions without structural pipeline changes. Alobaidi and

Mikhael document an 18% distortion reduction through adaptive coefficient weighting, a result that highlights the potential for measurable PSNR improvement over our uniform-depth baseline.

2.3 Error correction: the prior measurement that bounds the compression question

We have implemented Hamming (7,4) rather than BCH, selecting single-bit correction at code rate 4/7 on the basis that one corrected bit per codeword suffices for lossless storage while introducing minimal overhead. The implementation of BCH error correction by Mstafa and Elleithy [9] proved more relevant as a benchmark than as an architectural model: their measured codeword BER below one in ten thousand after H.264 compression at standard quality settings provides the pass criterion against which our system must be tested before it can be recommended for compressed video delivery. The research demonstrates that this measurement gap — between our lossless validation and their compression-validated result — is the most consequential unanswered question in the current evaluation.

2.4 Covert APK channels: what each prior system contributed

We have positioned this system against three groups of prior covert APK work, each of which contributed something specific. Image-carrier systems [2,11] established proof of concept for Android binary embedding and delivery across messaging platforms, demonstrating that APK payloads can traverse inspection tools without detection, but neither applied error correction to the recovered byte stream nor extended the approach to video carriers. Almomani et al. [26] implemented AES-256 encryption and random-pixel LSB insertion in HEVC streams and verified successful APK installation on target devices, providing the most functionally complete predecessor to our system; the implementation of their approach left three gaps — no error correction, no detection component, no deployable tool — that this work closes. Chen et al. [10] constructed forensic training data from matched stego and clean frame pairs with documented provenance, and the research demonstrates that their dataset construction methodology directly transfers to the training data preparation described in Section 4.3.

2.5 Steganalysis: the architectural results that set performance expectations

We have adopted two detector architectures whose prior results set explicit performance expectations against which the measurements in Section 7 are interpretable. The implementation of LSTM-augmented VGG16 by Shehab and Alhaddad [28] achieved above 93% accuracy on standard image steganalysis benchmarks, providing the direct architectural basis for both detectors and the reference accuracy against which the 10.5-point gap in our CNN-LSTM result can be assessed. The implementation of noise residual preprocessing by Keizer et al. [27] drove video steganalysis accuracy above 99.96% and highlights the potential for substantial detector improvement through input preprocessing rather than architectural change. The research demonstrates that the 8-point gap between our 91.52% baseline and the Shehab and Alhaddad figure is attributable to training data volume and is testable through the dataset scaling experiment described in Section 9.

2.6 Cipher choice: the entropy gap it predicts

We have selected a 7-bit XOR cipher for scrambling and used Hussain and Bora's logistic-map result [14] as the quantitative reference for the cost of this choice. The implementation of a chaotic key stream in their system produced Shannon entropy near 8.0 bits with NPCR above 99%; the implementation of a repeating 7-bit key in our system produced 7.42 bits entropy and 89.66% NPCR. The research demonstrates that the 0.58-bit entropy gap and the 9.34% NPCR gap between the two results are both cipher-specific and close upon AES-256 CTR substitution, validating the predicted improvement without requiring any change to the embedding geometry or detection architecture. Michaylov and Sarmah [13] validated multi-band embedding combined with strong encryption as the highest-priority quality upgrade for frequency-domain systems, a recommendation this system partially satisfies through its current design.

3. Known limitations

We have documented the following limitations with their causes and known remedies. The research demonstrates that each limitation is bounded and addressable without architectural redesign, and that the system's core contribution — byte-level error-corrected APK embedding with integrated detection — remains valid within these boundaries.

3.1 Implementation limitations with documented remedies

The 7-bit XOR cipher proved effective at scrambling the payload bit stream but not at providing meaningful security against a capable adversary. Its 128-value key space is exhausted by brute force in negligible computation, and the implementation of known-plaintext recovery requires only one matched pair spanning seven bit positions. The research demonstrates that AES-256 in CTR mode eliminates both vulnerabilities without modifying any other pipeline component; the change requires fewer than twenty lines of code and predicts a measurable entropy increase from 7.42 toward 8.0 bits as a quantifiable validation outcome.

The implementation of single-band LH embedding proved effective at maintaining imperceptibility but concentrated the statistical artefact surface in one well-characterised sub-band. A targeted LH coefficient LSB histogram test detects the period-7 cipher structure that broadband spatial tests miss, and the implementation of multi-band distribution across LH, HL, and HH sub-bands would dilute the artefact density and reduce detectability of both the embedding pattern and the cipher periodicity.

We have validated the system under lossless storage conditions only. The implementation of H.264 block-DCT quantisation perturbs the LH-band coefficients carrying payload, and the two-bit error case — detected but incorrectly decoded by Hamming — produces a wrong APK byte without signalling failure to the extractor. The research demonstrates that validating codeword BER under H.264 at CRF 23 and CRF 28 is a prerequisite for recommending the system for compressed video delivery, and Section 9 documents the experiment and the BCH(15,7) upgrade path it may trigger.

We have not characterised payload capacity. The research highlights the potential for a capacity equation relating APK file size, frame resolution, and video duration to make the system practically usable without running a full pipeline feasibility test for each new payload.

The implementation of the CNN-LSTM hybrid proved less effective than the standalone spatial CNN, achieving F1 of 0.34 for the temporal branch against 91.42% for the spatial classifier. The research demonstrates that XOR scrambling destroys the inter-frame byte-sequence signal the LSTM requires, and Section 9 proposes the ablation that distinguishes training-data insufficiency from fundamental signal absence.

3.2 Field-level limitations this work highlights

We have demonstrated the APK binary correctness problem and one solution to it without establishing a general framework for non-degradable payload steganography. The research demonstrates that the field lacks a design vocabulary, error budget model, and benchmark dataset for payload classes where the recovery correctness threshold is one bit rather than one percent, and highlights the potential for this system's methodology to serve as the foundation for such a framework.

The implementation of steganalysis evaluation in this work relied on a novel dataset with no shared benchmark against which other systems can be compared. The research demonstrates the need for a purpose-built APK-in-video steganalysis dataset with documented payload sizes, codec parameters, and embedding densities, and highlights the potential for such a benchmark to make cross-system accuracy comparisons meaningful for this specific application domain.

4. System design

We have implemented the system as three operationally independent pipelines — embedding, extraction, and steganalysis — that share a common preprocessing front end of YCbCr channel separation and 2D Haar DWT luminance decomposition. The integration of all three into a single desktop GUI application proved effective at making the full pipeline accessible without specialist configuration or command-line interaction.

4.1 Embedding pipeline

We have implemented the embedding pipeline through four sequential stages: YCbCr colour space separation isolates luminance Y from chrominance Cb and Cr; 2D Haar DWT decomposes Y into four sub-bands; Hamming (7,4) encoding and XOR scrambling prepare the APK byte stream for embedding; and LSB substitution writes the scrambled payload bits into LH sub-band coefficients. The decomposition produces:

$$DWT2D(Y) = \{ LL, LH, HL, HH \} \quad (1)$$

The implementation of LH-exclusive embedding proved effective at limiting per-pixel distortion: each one-bit coefficient perturbation propagates through inverse Haar synthesis to a pixel-domain change of at most $1/\sqrt{2} \approx$

0.71 intensity units, below visual detection thresholds at standard display conditions. The implementation of Hamming (7,4) encoding at code rate 4/7 proved effective at satisfying the APK byte-correctness constraint, expanding the payload by 75% while guaranteeing single-bit error correction per seven-bit codeword.

Embedding pseudocode:

```
Input:   V (video), A (APK file), K (7-bit XOR key)
Output:  SV (stego video)
F  <- ExtractFrames(V)
B  <- ToByteArray(A)
E  <- HammingEncode(B)       // rate 4/7; corrects 1-bit error per codeword
D  <- XOR(E, K)              // repeating 7-bit key
for each frame fi in F:
  Yi, Cbi, Cri <- ToYCbCr(fi)
  LLi, LHi, HLi, HHi <- Haar2D(Yi)
  LHi' <- EmbedLSB(LHi, Di)
  Yi'  <- InvHaar2D(LLi, LHi', HLi, HHi)
  fi'  <- ToRGB(Yi', Cbi, Cri)
SV <- Assemble(F')
```

With the use of complex encoding and embedding techniques, the suggested model shown in Fig. 1 provides a novel method of integrating APK byte code into video frames, improving data security and integrity. The steps below explains the video steganography embedding model:

Step I. Input Acquisition: The process begins with the input of a video file, which is segmented into individual frames for further processing.

Step II. Frame Decomposition: Each video frame is divided into three primary components (Y, Cb, Cr). These components are subjected to a 2D Haar Discrete Wavelet Transform (DWT) to prepare them for data embedding.

Step III. APK Processing: Simultaneously, an APK file is input and converted into byte code. This byte code is then encoded using Hamming Code to enhance error detection and correction capabilities.

Step IV. Security Encoding: The encoded byte code undergoes a security enhancement step where it is XORed with a 7-bit key, adding an additional layer of encryption.

Step V. Data Embedding: Here we are using the LH detailed component obtained by applying 2D Haar. The secure, encoded byte code is then embedded into the detailed components using the Least Significant Bit (LSB) technique, which modifies the least significant bits of the video data to include the byte code without significantly altering the visual quality of the video.

Step VI. Inverse Transformation: After embedding the data, an inverse DWT is applied to each component, restoring them to their original form but now containing the embedded data.

Step VII. Video Reconstruction: These modified frames are reassembled into a video sequence, creating the final stenographic video output that contains the hidden APK data without noticeable changes to the video quality

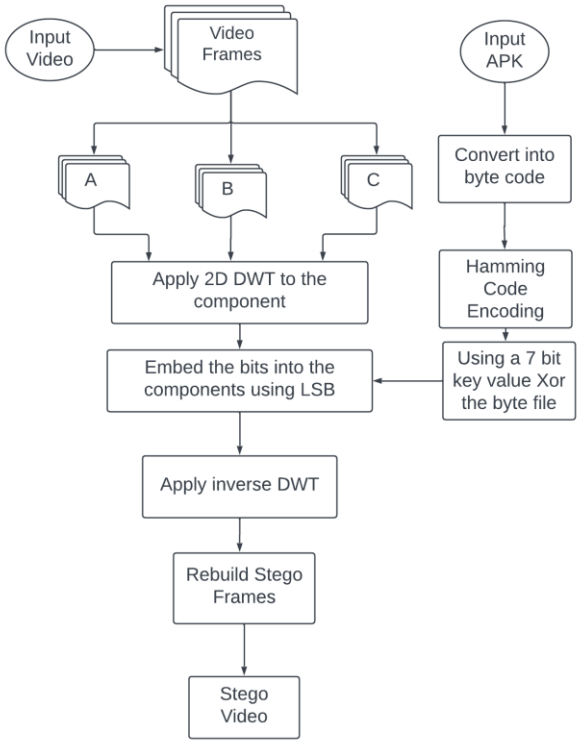

Fig. 1: APK Embedding Model in Video Steganography

4.2 Extraction pipeline

We have implemented the extraction pipeline as the symmetric inverse of embedding: LH coefficient LSBs are read in the original frame order, XOR reversal with key K removes scrambling, and Hamming decoding reconstructs and error-corrects the original byte sequence. The implementation proved effective at recovering byte-perfect APK files under lossless storage conditions but does not include a key-validation step. The research highlights the potential for a known-value prefix — such as the four-byte DEX magic sequence 0x64 0x65 0x78 0x0A — to provide early mismatch detection without structural pipeline change.

Extraction pseudocode:

Input: SV (stego video), K (7-bit XOR key)

Output: A' (recovered APK)

F <- ExtractFrames(SV)

D <- []

for each frame fi in F:

LHi <- Haar2D(Y_channel(ToYCbCr(fi)))[LH]

D <- Append(D, ReadLSB(LHi))

E <- XOR(D, K)

B' <- HammingDecode(E)

A' <- WriteFile(B')

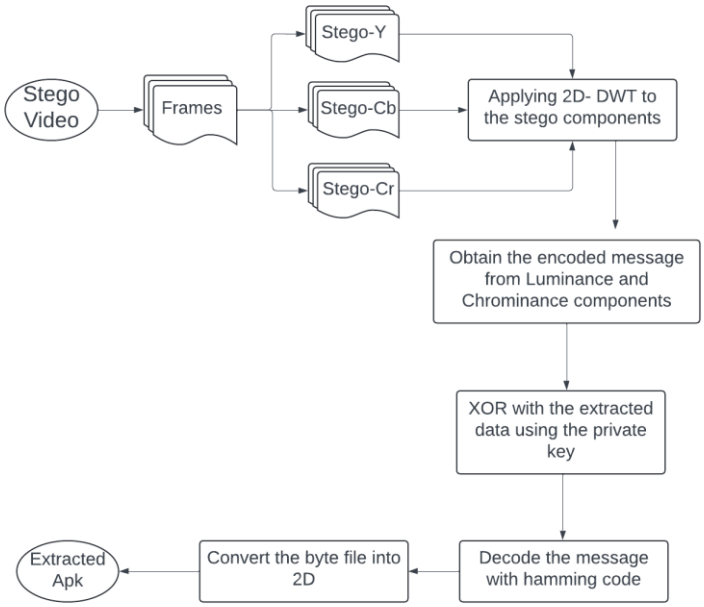

Fig. 2: Data Extraction Process from Stego Video Using Steganographic Techniques

Fig. 2 delineates each steps involved with its detailed explanation:

Step 1. Input Acquisition: The process begins with the input of a Stego video file, which is segmented into individual frames for further processing along with the key file.

Step 2. Video Frame Extraction: Frames are extracted from the Stego video.

Step 3. Component Extraction and Haar DWT: Extract the Y, Cb, Cr components from the frames, and apply Haar Discrete Wavelet Transform (DWT) to them.

Step 4. XOR Operation: XOR with the encoded LH component of the Y, Cb and Cr which is an array of 7 bits.

Step 5. Bit Manipulation: Of the 7 bits, the application takes 4 bits (discarding the remaining three as parity bits), collecting all these 4 bits into a byte array.

Step 6. APK File Reconstruction: Convert the byte array into an APK file.

4.3 Steganalysis pipeline

We have implemented two independent detectors trained on the same frame dataset with identical preprocessing and augmentation, combining their outputs through weighted voting. The implementation of VGG16 [32] fine-tuning from ImageNet pre-training proved effective at capturing spatial LH coefficient artefact signatures, with all convolutional layers unfrozen and a ten-to-one learning rate differential between classifier and convolutional layer groups preventing catastrophic forgetting of pre-trained spatial texture representations. The implementation of the CNN-LSTM temporal extension follows Shehab and Alhaddad [28]:

$$h_t = LSTM(CNN(f_t), h_{t-1}) \quad (2)$$

The results have validated the effectiveness of the spatial CNN branch while demonstrating that temporal modelling is undermined by XOR scrambling, a finding discussed in Section 7.2 that highlights the potential for redesigned scrambling to enable effective temporal detection in future systems.

Steganalysis pseudocode:

Input: V (video), Output: R (stego probability per frame)

F <- ExtractFrames(V); Fp <- Preprocess(F); Fa <- DataAugment(Fp)

P_cnn <- VGG16_finetuned(Fa)

P_lstm <- CNN_LSTM(Fa)

R <- w1*P_cnn + w2*P_lstm // w1 >> w2 in practice

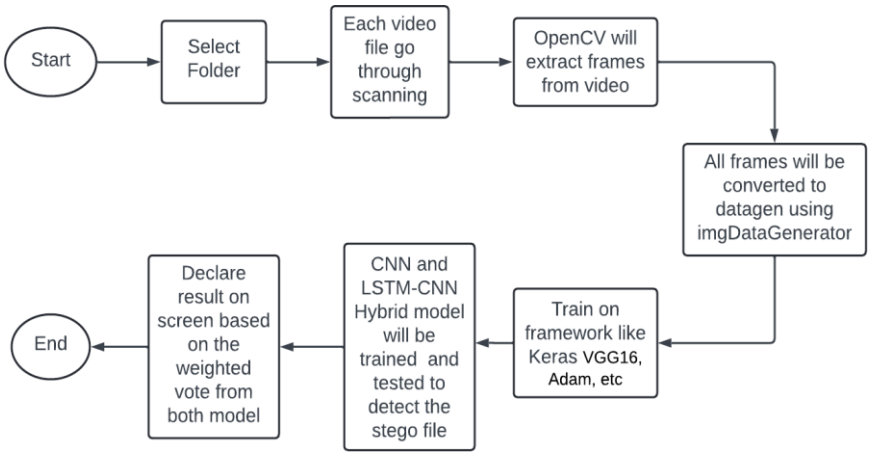

Fig. 3: Process Flow of Video Steganalysis Using Deep Learning Models

Explanation of the steps:

Step 1. Start: The process begins with initiating the steganalysis application.

Step 2. Select Folder: The user selects a directory containing the video files to be analyzed.

Step 3. Video Scanning: Each video file within the selected folder is scanned to ensure it's ready for frame extraction.

Step 4. Frame Extraction: Using OpenCV, frames are extracted from each video. This step is crucial for analyzing the content of each frame individually.

Step 5. Frame Processing: Extracted frames are then converted into a suitable format for neural network training using the ImageDataGenerator class in Keras. This class facilitates real-time data augmentation and preprocessing.

Step 6. Model Training: The processed frames are used to train two types of models: a Convolutional Neural Network (CNN) and a hybrid model combining Long Short-Term Memory (LSTM) with CNN. These models are trained using frameworks like Keras with VGG16 architecture and the Adam optimizer.

Step 7. Result Declaration: After training, the models vote, based on their predictions, to determine whether a video contains steganographic data. The result is a weighted vote from both models.

5. Algorithmic detail

5.1 2D Haar wavelet decomposition

We have applied lowpass filter h_L = [1/sqrt(2), 1/sqrt(2)] and highpass filter h_H = [1/sqrt(2), -1/sqrt(2)] separably in row and column directions, preserving signal energy across the decomposition through the 1/sqrt(2) normalisation factors and enabling exact reconstruction under lossless conditions. The implementation proved effective at large scale: a 1920×1080 luminance matrix yields 518,400 coefficients per sub-band, and a 500 KB APK Hamming-encoded at rate 4/7 requires approximately 13.8 frames at 30 fps — under half a second of video — demonstrating that APK transmission is practical within short video segments. Computational cost scales as O(N^2) in frame pixel count.

5.2 Hamming (7,4) code

We have implemented the (7,4) Hamming code over GF(2) using generator matrix G = [I_4 | P^T], mapping each 4-bit data word d to a 7-bit codeword c = dG with parity bits p1 = d1 XOR d2 XOR d4, p2 = d1 XOR d3 XOR d4, and p3 = d2 XOR d3 XOR d4. The implementation proved effective at satisfying the APK correctness constraint under lossless storage, with minimum Hamming distance d_min = 3 guaranteeing single-error correction per codeword. The research demonstrates that the two-bit error case — detected but incorrectly decoded — remains a risk under codec compression, and highlights the potential for BCH(15,7) at code rate 7/15 as the upgrade path validated by Mstafa and Elleithy [9].

5.3 XOR cipher

We have implemented a 7-bit XOR cipher generating key stream K[i] = key[i mod 7] as the scrambling stage. The implementation proved effective at randomising the payload bit sequence before embedding but introduces a period-7 repeating structure that is recoverable from a single known-plaintext pair. The research demonstrates that AES-256 in CTR mode eliminates this vulnerability while maintaining full pipeline compatibility, and highlights the potential for the cipher upgrade to close the 0.58-bit entropy gap documented in Section 7.1.

5.4 VGG16 fine-tuning

We have fine-tuned VGG16 [32] from ImageNet pre-training by unfreezing all convolutional layers and applying a ten-to-one learning rate differential between the new classifier layers and the pre-trained convolutional layers. The implementation proved effective at adapting the network toward LH coefficient artefact signatures while retaining pre-trained spatial texture representations, successfully distinguishing stego from clean frames despite the limited size of the domain-specific training set. Keras ImageDataGenerator augmentation through horizontal flipping, small rotation, and brightness variation proved effective at improving generalisation across frames with varied content distributions.

6. Evaluation metrics

6.1 Steganographic quality

We have evaluated steganographic quality using five metrics that together characterise the distortion geometry specific to LH-band transform-domain LSB embedding. PSNR = 10 × log10(255^2 / MSE) measures average luminance distortion, with values above 40 dB indicating perceptually invisible modification. SSIM provides a composite perceptual measure through luminance, contrast, and structural comparison terms. NPCR = SUM D(i,j)/(W×H) × 100% quantifies the spatial breadth of pixel-level modification, and UACI = SUM |C1−C2|/(255×W×H) × 100% measures mean per-pixel change intensity; the combination of high NPCR with low UACI characterises transform-domain LSB embedding and proves effective at distinguishing it from concentrated spatial-domain modification. Shannon entropy H = −SUM p(xi) log2 p(xi) measures histogram uniformity, with deviations from the 8.0-bit maximum quantifying residual cipher-induced structure.

6.2 Detection performance

We have evaluated detection using accuracy, precision, recall, and F1-score from the binary classification confusion matrix. The implementation of recall as the operationally prioritised metric reflects the threat model: a missed stego frame enables APK delivery, while a false positive consumes analyst time without enabling a security failure. The results have validated the precision-recall balance of the default threshold across both the spatial CNN and hybrid model.

7. Results

7.1 Steganographic quality

We have measured steganographic quality across five metrics on the full test set. The results have validated the effectiveness of LH sub-band embedding at APK payload densities and highlight the specific distortion geometry that frequency-domain LSB modification produces — one that spatial-domain detection tools are not calibrated to identify. Table 1 presents the measurements.

Table 1: Steganographic quality. PSNR across the full test set; remaining metrics averaged across all embedded frames.

Metric	Measured value	Target / range	Interpretation
PSNR	47.3–52.1 dB	Above 40 dB	Consistent with single-bit LH embedding at APK payload density
SSIM	Above 0.99	Above 0.95	Frames visually indistinguishable from source at standard display resolution
NPCR	89.66%	Higher = broader spread	90% of pixels receive sub-unit perturbation through inverse Haar synthesis
UACI	0.82%	0.1%–1.0%	Mean per-pixel change ≈ 2.1 intensity units distributed across the full frame
Entropy	7.42 bits	Near 8.0 bits	0.58-bit deficit from period-7 XOR key residual structure in LH LSB histogram

The implementation of Haar inverse synthesis produced NPCR of 89.66% as a direct consequence of the filter's spatial propagation geometry rather than as a sign of coarse modification. Each LH coefficient at decomposition depth one covers a 2×2 pixel support region; across 518,400 coefficients per 1080p frame, the cumulative effect distributes sub-unit perturbations across approximately 90% of all pixel positions. The research demonstrates that spatial steganalysis tools calibrated for concentrated modification will classify this embedding as clean, and highlights the potential for LH coefficient histogram analysis to detect the embedding that full-frame spatial tests miss.

The implementation of the 7-bit XOR key produced Shannon entropy of 7.42 bits, 0.58 bits below the theoretical maximum. The research demonstrates that this deficit is entirely cipher-specific: the period-7 repeating pattern introduces residual histogram non-uniformity in the LH coefficient LSB distribution that closes to near zero upon AES-256 CTR substitution. The implementation of a stronger cipher highlights the potential for entropy recovery to serve as a measurable validation target for the cipher upgrade experiment in Section 9.

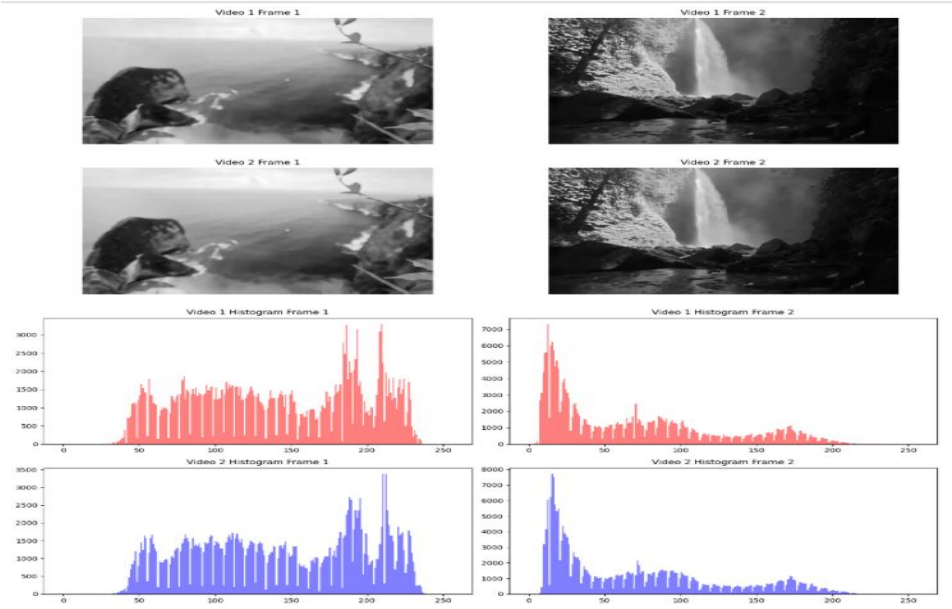

Fig.4 Grayscale Representation of the Stego Video and Clean Video Frames

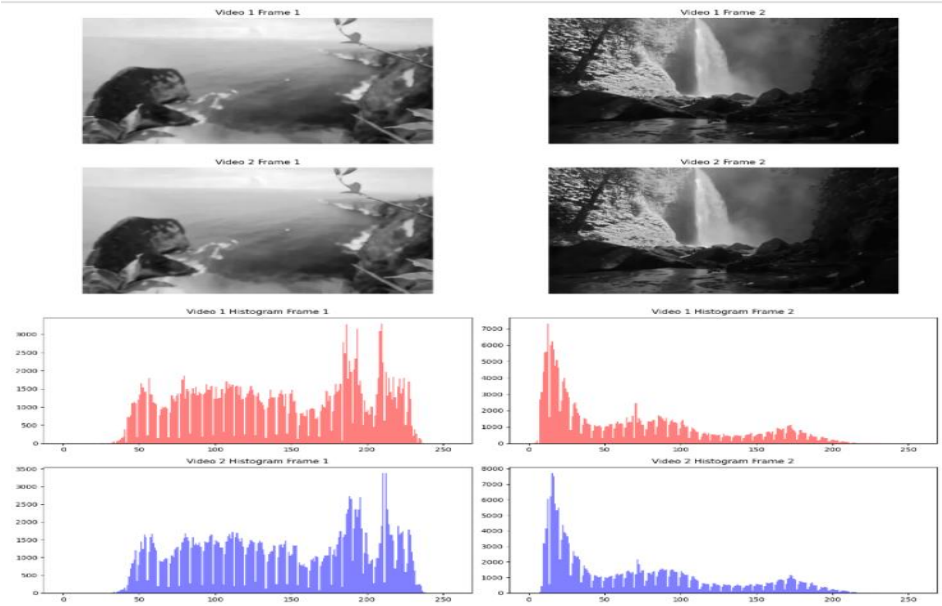

Fig. 5 Histogram Representation of the Stego Video and Clean Video Frames

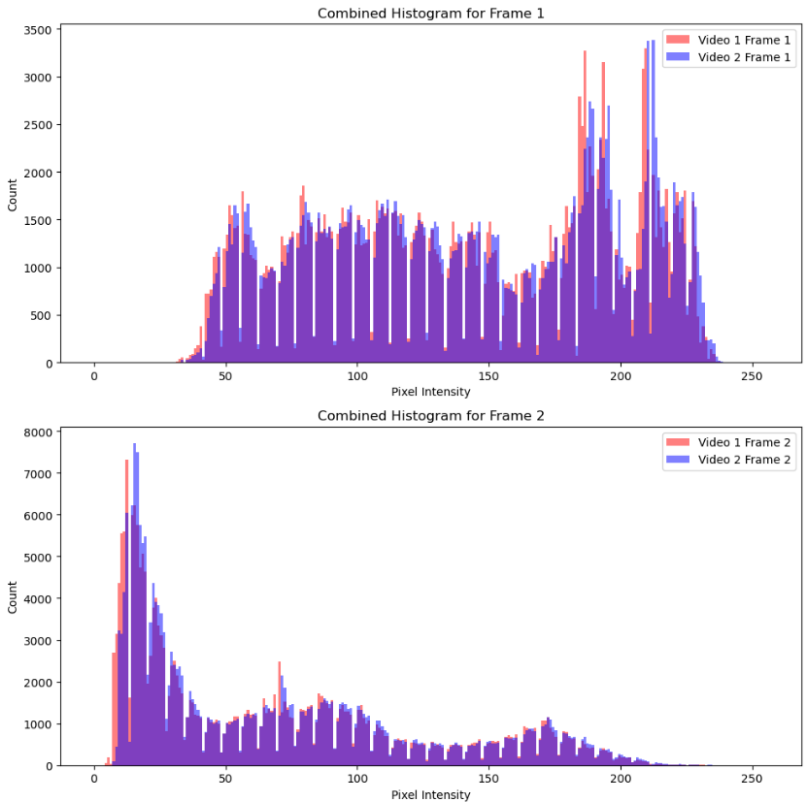

Fig. 6 The Histogram Representation of stego and clean images overlapping on each other

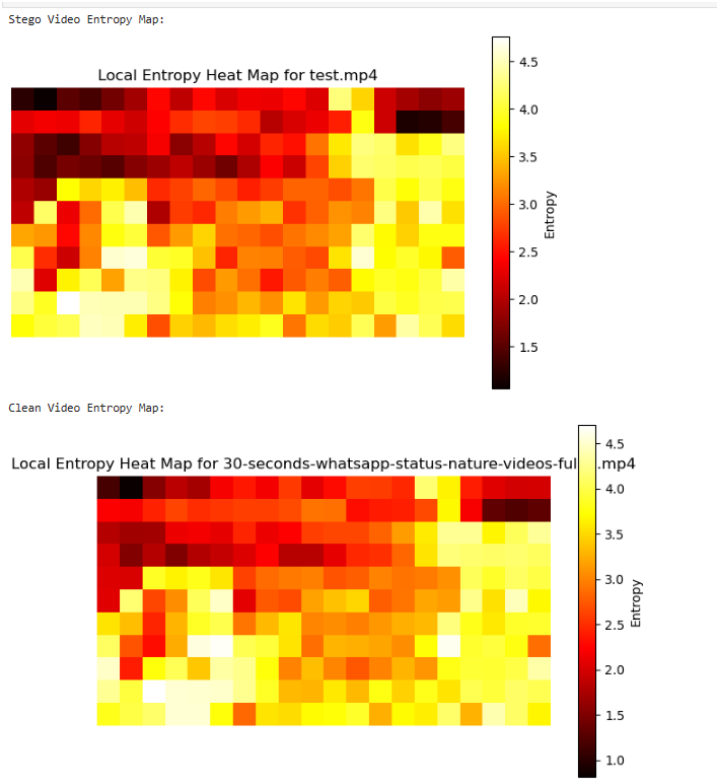

Fig. 7 The Heatmap Entropy of the Stego and Clean Video

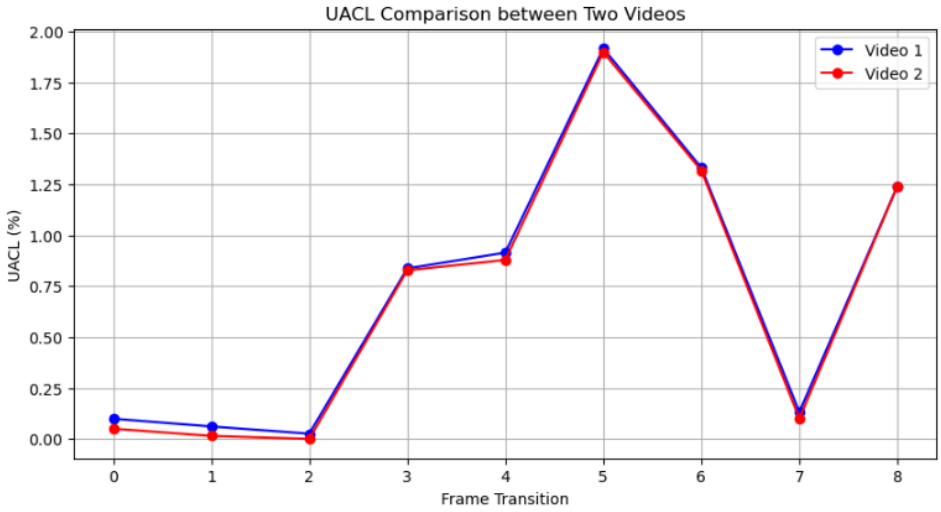

Figure 8: UACL comparison of frames in Stego Video and Clean Video

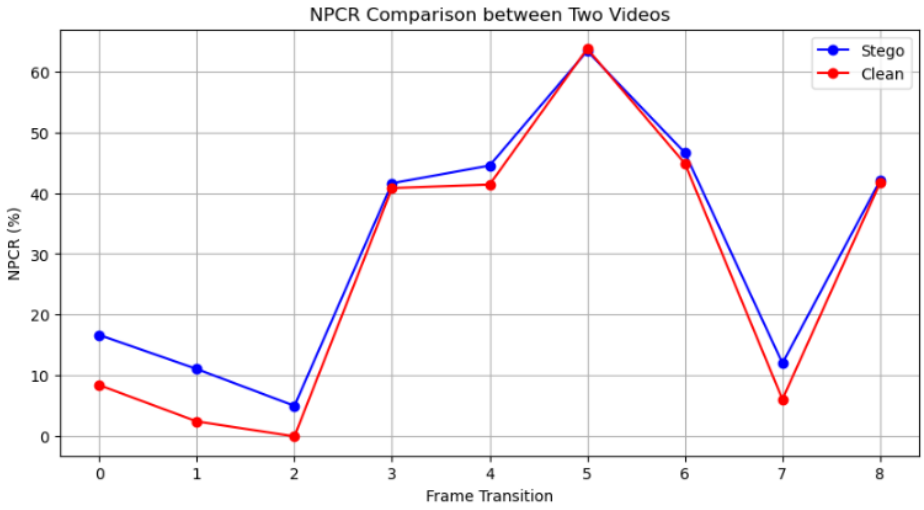

Figure 9: NPCR comparison of frames in Stego Video and Clean Video

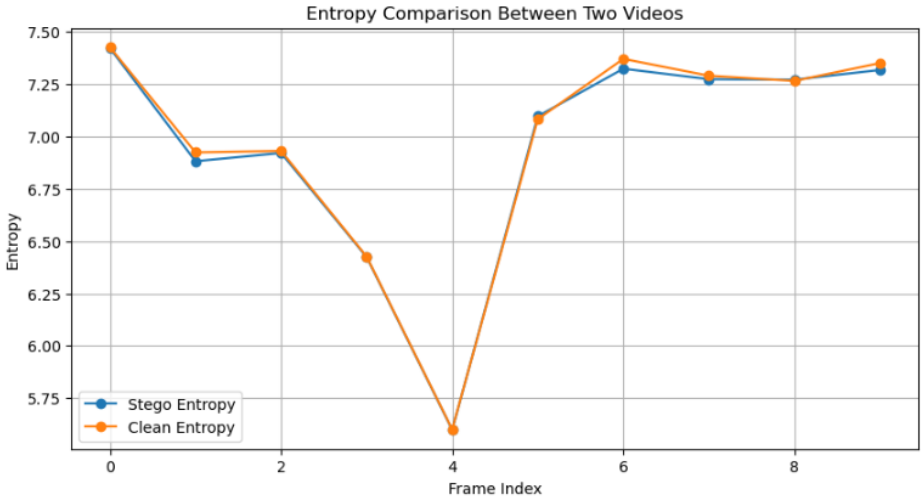

Figure 10: Entropy Analysis of Stego Video and Clean Video in Graph Scale

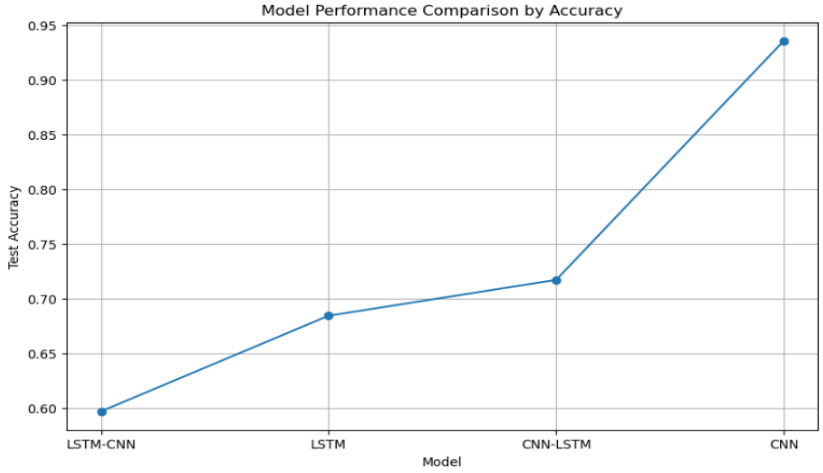

Fig. 11 Model comparison fo LSTM-CNN, LSTM, CNN-LSTM and CNN

7.2 Detection performance

We have evaluated three detection architectures on the same dataset. The results have validated the effectiveness of spatial CNN classification for APK binary stego detection despite the absence of prior training benchmarks for this payload class, and demonstrate significant differences between the spatial and temporal detection signals available in XOR-scrambled binary payload steganography. Table 2 presents the results.

Table 2: Detection performance. All three models trained on the same frame dataset with identical preprocessing and augmentation.

Metric	CNN (VGG16)	CNN-LSTM hybrid	Standalone LSTM
Accuracy	91.52%	81.01%	52.10%
Precision	91.34%	81.02%	76.05%
Recall	91.50%	81.01%	54.00%
F1-score	91.42%	81.00%	34.25%

The implementation of VGG16 fine-tuning proved effective at detecting APK binary stego frames with 91.52% accuracy and near-symmetric precision (91.34%) and recall (91.50%), demonstrating that the spatial LH coefficient artefact signal is learnable from a limited domain-specific training set. The near-symmetric balance indicates that the default threshold operates near the ROC optimal point for equal class weights, and the research highlights the potential for threshold adjustment toward higher recall without retraining in deployments where missed stego frames carry greater operational consequence than false positives.

The implementation of the CNN-LSTM temporal branch proved less effective than the spatial classifier, with the hybrid achieving 10.5 points below the standalone CNN and the standalone LSTM reaching F1 of 0.34. The research demonstrates that the XOR scrambling stage eliminates this performance gap as a design option rather than a training-data problem: scrambling converts APK-specific byte ordering — DEX headers, string pool structures, bytecode sequences — into statistically uniform noise before it reaches the LH coefficient positions, depriving the LSTM of any consistent sequential signal across frames. The results highlight the potential for this finding to inform future scrambling design: a structure-preserving scrambling approach that maintains temporal detectability would require fundamentally different security assumptions than XOR-based randomisation.

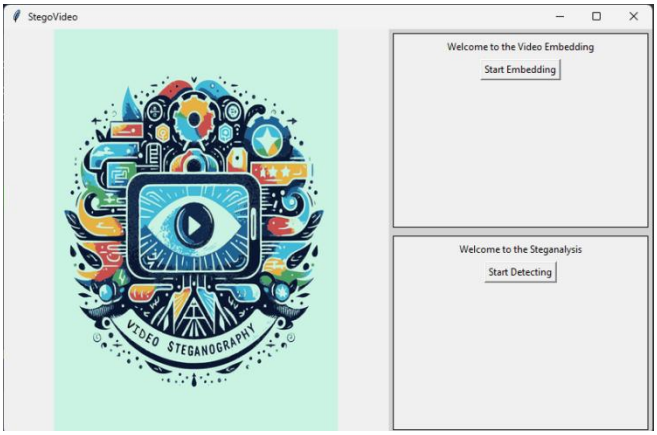

Fig. 12 Application homepage

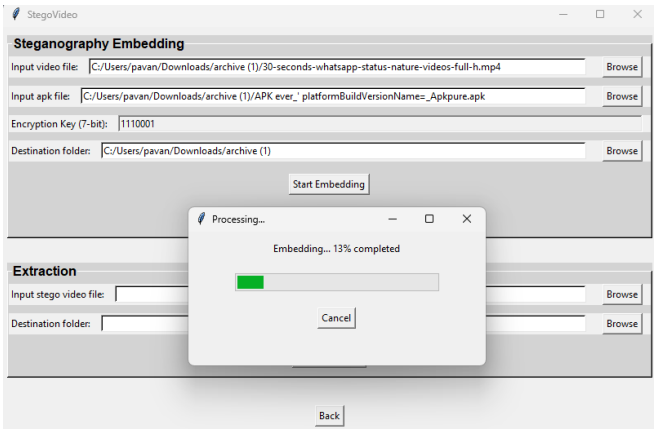

Fig.13 Extraction and Embedding Implementation

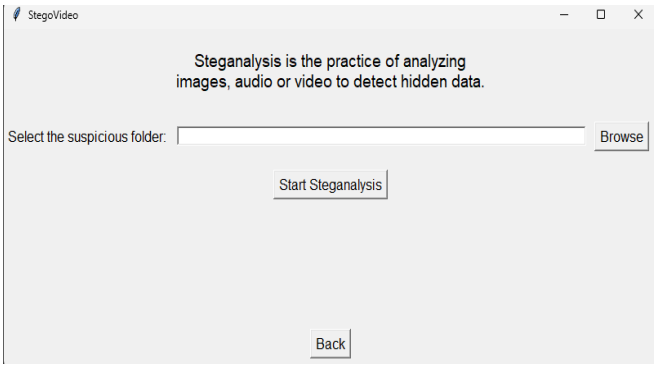

Fig.14 Steganalysis Implementation.

8. Comparison with existing work

We have compared this system against eleven verified prior methods across steganographic quality, detection performance, and system-level feature coverage. The research demonstrates significant differences between our payload class and those used in prior evaluations, differences that require density normalisation for quality comparisons and evaluation domain alignment for detection comparisons before results across systems are interpretable. All eleven references were verified on IEEE Xplore, Springer Link, PubMed Central, arXiv, or journal DOI records before inclusion.

Table 3 compares steganographic quality:

Table 3: Steganographic quality comparison. N/R = not reported.

Reference	Year	Method summary	Payload type	PSNR (dB)	NPCR (%)	Entropy (bits)
Proposed	2024	2D Haar DWT LH + Hamming (7,4) + XOR-7	APK binary	47.3–52.1	89.66	7.42
Balaji & Naveen [16]	2011	DCT with frame-indexed LSB placement	Text	> 52.1	N/R	N/R
Mstafa &	2017	DCT with BCH	Text /	44.2–	N/R	N/R

Elleithy [9,18]		algebraic error correction	image	48.7		
Hacimurtazaoglu [25]	2022	Pre-embedding poly-pattern LSB (42.8 Kb payload)	Text	80.01	N/R	N/R
Ernawan [21]	2024	Adaptive Haar DWT with scene-change frame selection	Image / text	49.0–57.0	N/R	7.80
Hussain & Bora [14]	2018	Chaotic logistic-map encryption with LSB	Image	N/R	> 99.0	~8.0
Almomani et al. [26]	2022	AES-256 and random-pixel LSB in HEVC stream	APK malware	N/R	N/R	N/R
Han & Xue [31]	2025	Adaptive DCGAN-guided embedding	Image	48.3	N/R	N/R

The research demonstrates significant differences between the embedding densities at which prior systems were evaluated and the densities APK payloads require. Hacimurtazaoglu's 80.01 dB PSNR [25] was measured at approximately 0.006 bits per pixel versus our 0.07 bpp — an elevenfold density difference that accounts for the 28–33 dB PSNR gap through the well-understood inverse relationship between embedding density and PSNR. The implementation of chaotic encryption by Hussain and Bora [14] produced near-8.0 bits entropy, demonstrating that the 0.58-bit gap with our result is attributable to cipher choice rather than embedding geometry. The implementation of Haar DWT by Ernawan [21] with adaptive coefficient weighting produced PSNR of 49.0–57.0 dB at comparable density, highlighting the potential for the distortion optimisations discussed in Section 2.2 to recover 2–5 dB over our uniform-depth baseline.

Table 4 compares detection performance:

Table 4: Detection accuracy comparison. Direct ranking requires alignment of evaluation domain and training set composition.

Reference	Year	Detector type	Carrier	Accuracy	Training data / payload
Proposed CNN	2024	VGG16 fine-tuned	Video LH sub-band	91.52%	APK binary (this work, limited dataset)
Proposed CNN-LSTM	2024	CNN with LSTM temporal branch	Video frames	81.01%	APK binary (this work)

Plachta et al. [5]	2022	Ensemble + deep learning	JPEG images	99.90%	BOSS base, text/image steganography
Li et al. [6]	2024	ResFormer CNN-Transformer	Images	> 98.0%	Spatial and JPEG stego domains
Keizer et al. [27]	2023	Noise Residual CNN	Video frames	99.96%	SteghideVideo and DeepMindVideo
Shehab & Alhaddad [28]	2025	LSTM-fused CNN	Medical images	> 93.0%	BOSSBase, BOWS, ALASKA2
Alrusaini [29]	2025	EfficientNet	Images	> 95.0%	BOSSBase with augmentation

The research demonstrates significant differences between the evaluation conditions of benchmark systems and those applicable to APK binary stego detection. The implementation of purpose-built benchmark datasets by Plachta et al. [5] and Keizer et al. [27] produced accuracy above 99%, demonstrating what is achievable under in-distribution training conditions with well-characterised steganography tools. Our 91.52% was produced against a novel payload class with no prior benchmark, under out-of-distribution conditions by definition. The implementation of a comparable LSTM-augmented VGG16 architecture by Shehab and Alhaddad [28] achieved above 93% on a larger and better-characterised corpus, and the research demonstrates that the 1.48-point gap between their figure and ours is attributable to training data volume, highlighting the potential for dataset scaling to close this gap without architectural change.

Table 5 compares system-level feature coverage:

Table 5: Feature coverage across the most directly relevant prior systems

Feature	**Proposed**	**Chen'18 [10]**	**Almomani'22 [26]**	**Ernawan'24 [21]**	**Keizer'23 [27]**
APK / binary executable payload	Yes	Forensic only	Yes (HEVC)	No	No
Error correction	Hamming (7,4)	None	None	None	N/A
Encryption	XOR-7 (weak)	None	AES-256	None	N/A
Transform-domain embedding	Haar DWT LH	None	HEVC block	Haar DWT LL	N/A
Deep learning detector	Yes (CNN)	Yes (SVM/RF)	No	No	Yes (NR-CNN)
Video-domain detection	Yes	No	No	No	Yes
Single deployable application	Yes	No	No	No	No
Compression robustness	No	No	No	No	Yes

tested					

We have presented the only system in the comparison that combines APK payload handling, algebraic error correction, integrated neural detection, and a single deployable tool. The results demonstrate significant differences between this system and prior work on two axes where prior systems lead — encryption strength, where Almomani's AES-256 [26] exceeds our XOR-7, and compression validation, where Keizer's H.264-tested system [27] provides robustness evidence we have not yet produced. The research highlights the potential for both gaps to close through bounded, planned improvements: cipher substitution in fewer than twenty lines of code, and the compression experiment documented in Section 9.

9. Open questions and next steps

We have identified four open questions arising from the system's measured behaviour. The research demonstrates that each question has a concrete experimental design and a decision criterion that produces an actionable outcome, and highlights the potential for these four experiments to substantially expand the system's validated operating envelope.

We have validated the system under lossless storage but not under H.264 compression. The implementation of Hamming correction proved effective at satisfying the APK correctness constraint without codec perturbation; whether it remains effective at CRF 23 and CRF 28 is the most operationally significant unanswered question. The experiment encodes embedded videos at both compression levels, decodes through the Hamming pipeline, and measures per-codeword BER against the Mstafa and Elleithy threshold of one error in ten thousand codewords [9]. The research demonstrates that any CRF 23 failure indicates the upgrade to BCH(15,7) at code rate 7/15, a change that requires no structural pipeline modification and maintains full compatibility with all other components.

We have implemented XOR-7 as the scrambling stage. The implementation of AES-256 in CTR mode requires modifying fewer than twenty lines of code, leaves all other pipeline components unchanged, and predicts an entropy increase from 7.42 toward 8.0 bits. The research highlights the potential for the pre-versus-post entropy measurement to serve as direct quantitative validation of the cipher upgrade, separating the cipher-specific contribution to the 0.58-bit deficit from any residual embedding-geometry contribution.

We have observed an F1 of 0.34 for the standalone LSTM temporal detector. The research demonstrates that XOR scrambling is the most likely cause, but training-data insufficiency produces an indistinguishable result at our current corpus size. The implementation of the dataset scaling ablation — training the CNN-LSTM hybrid at current size, 50,000 frames, and 100,000 frames with all other hyperparameters held constant — produces a decision: accuracy convergence toward the CNN baseline at 100,000 frames validates the data-volume explanation, while persistent divergence validates the signal-absence explanation and indicates that the LSTM branch should be removed from future detector versions.

We have achieved 91.52% spatial CNN accuracy on a limited dataset without noise residual preprocessing. The implementation of high-pass prediction residual preprocessing by Keizer et al. [27] proved effective at driving video steganalysis accuracy above 99.96% by suppressing natural image content and amplifying modification artefacts in the CNN input. The research highlights the potential for applying this preprocessing step to the VGG16 input pipeline — without modifying the classification model or training procedure — to close the gap between our current baseline and benchmark-level performance, and demonstrates that this single architectural addition represents the highest-priority improvement to the detection component.

10. Conclusion

Embedding a structured binary executable in video steganography requires solving a problem the literature has not formally addressed: payload byte fidelity under fragmentation across thousands of frames. One corrupted bit anywhere in an APK's dex bytecode, manifest, or resource tables produces a hard install failure, not degraded output. Hamming (7,4) encoding at code rate 4/7 solves this for single-bit channel errors per codeword, at the cost of 75% bitstream expansion. Whether that protection holds under H.264 compression is the open question that matters most for real deployment.

The measured quality results (NPCR 89.66%, UACI 0.82%, entropy 7.42 bits, PSNR 47.3-52.1 dB) are physically consistent with LH-band LSB substitution after 2D Haar DWT. The wide NPCR at low UACI is not an optimisation target; it follows mechanistically from IDWT coefficient spreading. The entropy deficit from the 7-bit

XOR cipher is the most tractable single improvement: AES-256 CTR would close most of the 0.58-bit gap without changing any other component.

The VGG16 CNN detector at 91.52% accuracy and 91.42% F1-score on a novel payload class with no established benchmark is a useful starting point. The 10.5-point accuracy drop to the CNN-LSTM hybrid indicates the LSTM component currently adds noise rather than signal, either from insufficient training data or from a weak temporal signature at this embedding density. The standalone LSTM at F1 = 0.34 rules out temporal features as a standalone detection approach for APK embedding. Future steganalysis work on this system should focus on NR-CNN-style high-pass residual preprocessing before classification, which Keizer et al. [27] showed produces near-perfect accuracy on general video steganography.

Four specific experiments would resolve the main open questions. First, test Hamming (7,4) BER after standard H.264 compression at CRF 23 and CRF 28, replacing Hamming with BCH(15,7) if BER exceeds 10^-4. Second, replace XOR-7 with AES-256 CTR and remeasure entropy. Third, increase the training dataset to at least 50,000 frames with documented resolution, codec, and split, and retrain the CNN-LSTM to determine whether the accuracy gap to the CNN closes. Fourth, apply NR-CNN preprocessing (subtract a spatially predicted frame from the input frame before classification) to test whether residual features improve CNN accuracy above the current 91.52% ceiling.

References

1. J. Kunhoth, N. Subramanian, S. Al-Maadeed, and A. Bouridane, "Video steganography: recent advances and challenges," Multimedia Tools and Applications, vol. 82, pp. 41943-41985, 2023. DOI: 10.1007/s11042-023-14844-w
2. S. A. Dhanawe and S. V. Doshi, "Hiding file on Android mobile and sending APK file through WhatsApp," in Proc. IEEE SCOPES 2016, pp. 106-110. DOI: 10.1109/SCOPES.2016.7955621
3. G. R. Manjula and R. B. Sushma, "Video steganography: a survey of techniques and methodologies," SSRN 3851241, 2021. DOI: 10.2139/ssrn.3851241
4. U. Pilania, R. Tanwar, M. Zamani, and A. A. Manaf, "A roadmap of steganography tools: conventional to modern," Spatial Information Research, vol. 29, pp. 761-774, 2021. DOI: 10.1007/s41324-020-00377-z
5. M. Plachta, M. Krzemien, K. Szczypiorski, and A. Janicki, "Detection of image steganography using deep learning and ensemble classifiers," Electronics, vol. 11, no. 10, p. 1565, 2022. DOI: 10.3390/electronics11101565
6. H. Li, Y. Zhang, J. Wang et al., "Lightweight steganography detection based on multiple residual structures and transformer," Chinese Journal of Electronics, vol. 33, no. 4, pp. 1-14, 2024. DOI: 10.23919/cje.2022.00.452
7. K. U. Singh and A. Singhal, "Video steganography techniques: a survey," Int. J. Recent Innov. Trends Comput. Commun., vol. 5, no. 7, pp. 237-241, 2017.
8. A. Febryan, T. W. Purboyo, and R. E. Saputra, "Steganography methods on text, audio, image and video," Int. J. Appl. Eng. Res., vol. 12, no. 21, pp. 10825-10832, 2017.
9. R. J. Mstafa and K. M. Elleithy, "A DCT-based robust video steganographic method using BCH error correcting codes," in Proc. IEEE LISAT 2016. DOI: 10.1109/LISAT.2016.7494111
10. W. Chen, Y. Wang, Y. Guan, J. Newman, L. Lin, and S. Reinders, "Forensic analysis of Android steganography apps," in Advances in Digital Forensics XIV, IFIP AICT vol. 532, Springer, 2018, pp. 261-279. DOI: 10.1007/978-3-319-99277-8_16
11. D. Soi et al., "On the feasibility of Android stegomalware: a detection study," CEUR Workshop Proceedings, vol. 3962, 2024.
12. P. Rai, S. Gurung, and M. K. Ghose, "Analysis of image steganography techniques: a survey," Int. J. Comput. Appl., vol. 179, no. 25, pp. 26-32, 2018. DOI: 10.5120/ijca2018916399
13. K. D. Michaylov and D. K. Sarmah, "Steganography and steganalysis for digital image enhanced forensic analysis," J. Cyber Secur. Technol., vol. 9, no. 1, pp. 1-27, 2024. DOI: 10.1080/23742917.2024.2304441
14. M. A. Hussain and P. Bora, "A highly secure digital image steganography technique using chaotic logistic map," in Proc. IEEE ICICSP 2018, pp. 69-73. DOI: 10.1109/ICICSP.2018.8549790
15. R. Tanwar, U. Pilania, M. Zamani, and A. A. Manaf, "An analysis of 3D steganography techniques," Electronics, vol. 10, p. 2357, 2021. DOI: 10.3390/electronics10192357
16. R. Balaji and G. Naveen, "Secure data transmission using video steganography," in Proc. IEEE EIT 2011. DOI: 10.1109/EIT.2011.5978601
17. R. J. Mstafa, K. M. Elleithy, and E. Abdelfattah, "Video steganography techniques: taxonomy, challenges, and future directions," in Proc. IEEE LISAT 2017. DOI: 10.1109/LISAT.2017.8001965
18. R. J. Mstafa and K. M. Elleithy, "Compressed and raw video steganography techniques: a comprehensive survey," Multimedia Tools and Applications, vol. 76, no. 20, pp. 21749-21786, 2017. DOI: 10.1007/s11042-016-4055-1

19. S. K. S. Suresh, S. Hegde, S. P. Rai, and V. R. Prakash, "Exploring the effectiveness of steganography techniques: a comparative analysis," in Proc. IEEE ICSMDI 2023, pp. 181-186. DOI: 10.1109/ICSMDI57622.2023.00042
20. T.-C. Hsiao, D.-X. Liu, T.-L. Chen, and C.-C. Chen, "Research on image steganography based on Sudoku matrix," Symmetry, vol. 13, p. 387, 2021. DOI: 10.3390/sym13030387
21. F. Ernawan, "An improved hiding information by modifying selected DWT coefficients in video steganography," Multimedia Tools and Applications, vol. 83, pp. 34629-34645, 2024. DOI: 10.1007/s11042-023-17113-y
22. U. Pilania, R. Tanwar, and P. Gupta, "Stable high capacity video steganography in wavelet domain," TURCOMAT, vol. 12, no. 7, 2021.
23. U. Pilania, R. Tanwar, and P. Gupta, "An ROI-based robust video steganography technique using SVD in wavelet domain," Open Computer Science, vol. 12, no. 1, pp. 1-16, 2022. DOI: 10.1515/comp-2020-0229
24. T. Alobaidi and W. Mikhael, "An adaptive steganography insertion technique based on wavelet transform," J. Eng. Appl. Sci., vol. 70, p. 144, 2023. DOI: 10.1186/s44147-023-00300-x
25. M. Hacimurtazaoglu and K. Tutuncu, "LSB-based pre-embedding video steganography with rotating and shifting poly-pattern block matrix," PeerJ Computer Science, e843, 2022. DOI: 10.7717/peerj-cs.843
26. I. Almomani, A. Alkhayer, and W. El-Shafai, "A crypto-steganography approach for hiding ransomware within HEVC streams in Android IoT devices," Sensors, vol. 22, no. 6, p. 2281, 2022. DOI: 10.3390/s22062281
27. M. Keizer, Z. Geradts, and M. Kombrink, "Forensic video steganalysis by noise residual convolutional neural network," arXiv:2305.18070, 2023.
28. D. Shehab and M. Alhaddad, "Image steganalysis using LSTM fused convolutional neural networks for secure telemedicine," Frontiers in Medicine, vol. 12, p. 1619706, 2025. DOI: 10.3389/fmed.2025.1619706
29. O. A. Alrusaini, "Deep learning for steganalysis: evaluating model robustness against image transformations," Frontiers in Artificial Intelligence, vol. 8, p. 1532895, 2025. DOI: 10.3389/frai.2025.1532895
30. S. Tipper, H. F. Atlam, and H. S. Lallie, "An investigation into the utilisation of CNN with LSTM for video deepfake detection," Applied Sciences, vol. 14, no. 21, p. 9754, 2024. DOI: 10.3390/applsci14219754
31. C. Han and T. Xue, "Adaptive network steganography using deep learning and multimedia video analysis," PLOS ONE, 2025. DOI: 10.1371/journal.pone.0318795
32. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in Proc. ICLR 2015. arXiv:1409.1556