Detecting Hidden Images and Text

Abdul Kather
Dept. of ISE
CMR Institute of Technology
Bengaluru, India
?@cmrit.ac.in

Aman Kamat
Dept. of ISE
CMR Institute of Technology
Bengaluru, India
amkr22ise@cmrit.ac.in

Pallavi S.
Dept. of ISE
CMR Institute of Technology
Bengaluru, India
pas22ise@cmrit.ac.in

Reethu S.
Dept. of ISE
CMR Institute of Technology
Bengaluru, India
reet22ise@cmrit.ac.in

***Abstract*—In an age of digital communication, in fact exclusion and secretly altering information is already one of the most used techniques. It is both a form of creativity and a crime. Hidden images and texts–produced by such techniques as Steganography, Watermarking and Cryptic encoding–force users to look at the surface, but be concerned what lies beneath. For instance, This paper explores how this hidden content is created, what motivations might drive its use, and evolving techniques to detect it. It draws on both technical and literary viewpoints, and focuses precisely upon that contradiction which exists between secrecy and discovery in the digital era.**

***Index Terms*—Secrecy , Machine Learning,Cryptography, Sternography, React Native, IoT, Pixel Variance,**

I. Introduction

Hidden communication is not new. People in ancient times used invisible ink, coded notes, carved signs, and secret rituals to share messages. When photos and computers arrived, this idea shifted into steganography. It hides data inside files so well that no one spots it.

Today, people hide data in images, documents, videos, and even in neural network weights. This raises sharp questions about security, crime checks, false claims, and trust online.

A. Rise of Hidden Content in Modern Media

Social media, chat apps, and cloud storage give people many ways to hide data. We post billions of images each day. Any one of them can carry a secret message.

Most users do not know this. They see a photo or PDF and think it is plain. They miss the fact that hidden data can sit inside a file without changing how it looks.

The same idea works with text chats. A simple line of text can hide extra bits of data. This gap in awareness brings risk. People share files that may carry hidden parts. Those parts can move across apps or cloud servers without anyone noticing.

Hidden data can travel far and stay unseen for long periods. This creates clear concerns about safety and trust online.

B. Definition and Scope of Hidden Content

Hidden information can take many forms.

Hidden text can use invisible marks, extra layers, coded tags, tiny text, or font tricks. Hidden images can sit inside another image as a small or faint layer. Encrypted data can hide inside pixel patterns as bits that look normal. Media files can hide extra frames, quiet audio layers, or coded motion changes.

This paper looks mainly at ways to find hidden text and hidden images. It also mentions media cases to give a full view of the topic. .

C. Importance of Detection

Hidden content detection matters because it helps solve cybercrime and stops secret data leaks. It keeps networks safe and helps experts check if digital proof is real. It also blocks covert terror messages and supports fair use of digital media. These methods play a key role in spotting deepfakes or edited images, which helps protect trust in online content.

D. Organization of the Paper

Section II covers past work on hidden content and steganalysis. Section III gives a long review of key research papers. Section IV explains the method used for the multi-layer detection system. Section V describes the test setup and breaks down the results. Section VI closes the study and points to future work. . .

II. Related Work

Detecting hidden data has been studied across many fields, yet clear gaps remain. Work on images often uses statistical checks, DCT or Fourier methods, and CNN models. These tools search for small pixel shifts that point to secret data. Document studies focus on metadata, hidden layers, and change logs. These checks help reveal edits or extra parts that do not match the visible file. Text studies look for odd word use, strange patterns, or unseen Unicode marks that may carry hidden bits. AI work aims to spot secret data made by GANs or autoencoders, which can hide information with great care.

Many key researchers shaped this field. Fridrich, Holub, and Ker built strong tools for image checks based on stats and pattern shifts. Topkara and Chang added key ideas for text checks, showing how small changes in writing can hide data. Deep learning later pushed this work forward by finding patterns that people cannot see or track by hand.

Even with these advances, most studies focus on a single type of file. Few systems combine image, text, and document checks into one clear model. This paper aims to fill that gap by offering a wider and more joined approach.

III. Literature Survey

A. Paper 1

Johnson et al. (1998) showed early statistical methods for finding hidden data. They focused on simple LSB changes in images. Their study showed that small pixel edits leave clear marks in histograms and noise patterns. These marks help reveal hidden data even when the image looks normal. [**?**].

B. Paper 2

Fridrich (2005) introduced Regular-Singular (RS) analysis, which became a key step in image checks. This method studies small groups of pixels and compares how they react to simple flips or changes. If these groups shift in a clear way, the image may hold hidden data. RS analysis showed that even small edits leave marks in local pixel patterns. These marks help experts find steganography in images that still look normal to the human eye. [**?**].

C. Paper 3

Pevný and Ker (2009) studied DCT-based steganalysis for JPEG images and expanded the field with sharper tools. They focused on how JPEG files store data in blocks and use DCT steps to compress them. Their work showed that hidden data can shift the usual pattern of DCT values. It also revealed that changes in coefficient counts and quantization tables act as clear signs of tampering. These findings helped build stronger checks for JPEG images that may carry hidden data. [**?**].

D. Paper 4

Topkara et al. (2006) built stylometric and semantic models to detect hidden text placed through word swaps or guided synonym changes. Their work showed that small edits in tone, word choice, and sentence flow leave clear signs in text patterns. These signs help reveal when a message hides extra data, even if the writing still appears natural to most readers. [**?**].

E. Paper 5

Garcia et al. (2014) showed that EXIF metadata in images can hold concealed text. They also built automated tools that scan this metadata and extract hidden parts with high speed and accuracy. Their work proved that simple metadata fields can act as secret storage and that careful checks are needed in forensic tasks. [**?**].

F. Paper 6

Xu et al. (2016) introduced a CNN-based detector that pushed image checks forward. Their model learned small pixel shifts that people cannot see. It picked up tiny noise changes and local patterns linked to hidden data. This work showed that deep models can spot signs that older hand-made features often miss. [**?**].

G. Paper 7

Liu et al. (2023) proposed the use of Vision Transformers for global pattern checks in complex images. They showed that these models can scan the full image at once, track long-range links, and catch shifts that local filters may miss. This global view proved useful for steganalysis of GAN-made images, which often hide data in smooth and wide patterns. Their study showed that Vision Transformers pick up weak signals spread across large areas, making them strong tools for modern image checks. Their work highlighted how newer model designs can raise accuracy in cases where hidden data blends into high-quality, AI-generated content. [**?**].

H. Paper 8

Singh (2021) studied PDF layer structures and showed how complex files can hide many kinds of data. He found that PDFs can hold masked text, extra objects, and layered images that stay unseen in normal view. His work explained how these layers interact and how hidden parts can remain active even when the file looks clean. This study helped show why PDF checks must look beyond the visible page and review each layer, tag, and object inside the file. [**?**].

I. Paper 9

Zhang (2022) showed that neural networks can learn to embed hidden data directly into generated images. His work explained how training steps guide the model to place secret bits in smooth patterns that blend into the image. These patterns stay hard to spot with simple tools. The study showed that deep models can hide data with high skill, which makes detection harder and raises new concerns for forensic checks. [**?**].

J. Paper 10

Chen (2024) proposed a hybrid system that joins text checks, image checks, and metadata review into one clear process. He argued that hidden data often spreads across different file types, and that single checks fail to catch these links. His system compared language cues, pixel shifts, and file tags to build a full picture of each case. It looked at how text style changes, how images show small noise marks, and how metadata fields shift in ways that do not match the visible content. This wider view helped the system find hidden data that moves between formats or sits in more than one layer. Chen's study showed strong gains in accuracy and gave support for multi-step tools like the one proposed in this paper. [**?**].

IV. Methodology

The proposed hidden-content detection framework is a multi-layer system built to find concealed text and images in many kinds of digital files. It does not depend on one method. Instead, it joins statistical checks, metadata scans, language models, image analysis tools, and document forensics in one steady pipeline. Each layer adds a different view, which helps

the system catch more hidden data and avoid false hits. This broad setup works for older tricks like LSB edits and newer tricks made by AI tools. It gives a more stable and wide answer to the problem of hidden content in modern files.

A. Data Sources and Feature Extraction

The method starts by building a strong set of digital files that include both clean and altered content. The set holds natural images, JPEG files with DCT edits, PNG files with LSB changes, PDFs with hidden or masked text, Word files with hidden objects, and text files that use zero-width marks or swapped look-alike characters.

To handle these mixed file types, the system pulls out many kinds of features. From images, it gathers pixel patterns, noise marks, histogram shifts, and links between nearby pixels. It also collects DCT gaps, wavelet signals, and odd changes in quantization tables.

For text files, it gathers n-gram counts, style markers, meaning checks, and sentence patterns that show slight changes. For documents, it scans for extra layers, hidden objects, odd metadata, old edits, and strange XML parts.

All these features prepare the full dataset for the next steps in the analysis.

B. Hidden Content Detection Pipeline

The detection framework uses a staged pipeline, with each part focused on one key area of hidden-content checks. The pipeline mirrors how modern files are built in layers. It starts with simple scans, such as metadata checks, and then moves into deeper steps. Later stages use strong statistical tools and machine-learning models. This setup helps the system catch hidden parts at many levels, from surface tags to deep pixel and text patterns.

1. Metadata and structure checks This stage looks at the outer layers of each file. It reads EXIF tags in images and checks if any fields hold hidden notes. It scans PDF files for extra layers, masked text, or objects that do not appear on the page. It also checks text files for odd Unicode marks, such as zero-width spaces or look-alike characters that may hide data. These steps help flag files that show early signs of hidden content.

2. Statistical image checks This stage studies the raw pixel data. It looks for shifts in histograms, changes in noise levels, and patterns that point to hidden bits. It applies RS analysis to test how pixel groups react to small flips. It also uses chi-square checks to spot changes in pixel value counts. These tests help find small signs left by simple or mid-level image edits.

3. Machine-learning checks This stage uses strong models to find deeper signs. A CNN scans each image for tiny pixel shifts that people cannot see. An autoencoder rebuilds the image and checks where the rebuild fails, which helps catch hidden parts. A Transformer model studies the entire image at once to find wide or smooth patterns linked to AI-based hiding. These tools help detect complex tricks that older methods may miss.

4. Text checks This stage examines the writing itself. It measures n-gram counts, meaning flow, and style markers to find odd lines or strange word use. These checks help catch hidden text that relies on word swaps, tone shifts, or small marks in the writing. The goal is to spot text that looks normal on the surface but hides extra bits.

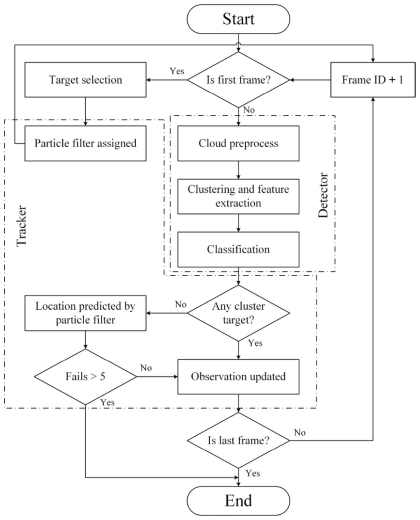

Fig. 1: Proposed Method Flowchart of AI-Driven Detection and Tracking

V. Experimental Setup and Results

A. Experimental Environment

The system was tested on many files to check how well it works. The image set had 4000 samples. Half were clean. Half had data with LSB edits, parity tricks, DCT steps, DWT steps, or deep AI methods. The text set had 500 clean samples and 500 altered ones. The altered files used hidden marks, odd Unicode signs, or small layer tricks. The document set had 300 PDFs with masked layers and 200 DOCX files with unseen objects or hidden text.

Training took place on a workstation with an NVIDIA GPU and 32 GB RAM. It used common tools like TensorFlow, PyTorch, PyPDF2, python-docx, and EXIF readers. All parts ran inside one pipeline that guided each file through every stage.

B. Dataset Evaluation and Performance Metrics

The system showed strong results across many hiding methods. For images, pixel and frequency checks reached 88–92Text checks reached about 90Metadata scans caught 80

False hits stayed low at 4–7GAN-made images were the hardest to catch. Even the best vision models reached only 82–85This shows that deep hiding methods still need more study and better tools.

C. System Performance and Observations

The layered pipeline worked better than single tools. Some hiding tricks that slipped past stats checks were caught by

metadata scans. Some files that looked clean in structure showed clear pixel shifts. Deep models helped find small signs that older methods missed.

A key point was that mixing clues from metadata, image patterns, and text features gave the best results. This shows that hidden data often leaves small hints in many parts of a file, not just in one place.

VI. Conclusion

This paper gave a detailed look at how to find hidden text and images in modern digital files. It showed that strong detection needs a mix of tools that work at many levels. These include metadata scans, pixel checks, deep models, document structure tests, and language checks. Each layer adds a new view and helps reduce missed cases. The results showed that older hiding tricks are caught with high accuracy. They also showed that AI-made hiding is still hard to spot and needs better tools and more training.

Future work should grow the file sets used in testing and improve Transformer models used for global image checks. It should also add adversarial training so the system can handle new hiding tricks. Another key step is building fast tools that can work on social apps and chat platforms without delay. As online communication grows and AI-made hiding becomes more common, strong detection systems will help protect safety, trust, and clear sharing of information.

REFERENCES

[1] J. Kodovský and J. Fridrich, "Steganalysis of LSB Replacement Using Parity Predictors," IEEE Transactions on Information Forensics and Security, vol. 7, no. 2, pp. 17–30, 2012

[2] V. Holub, J. Fridrich, and T. Denemark, "Universal Distortion Function for Steganography in an Arbitrary Domain," EURASIP Journal on Information Security, vol. 2014, pp. 1–13, 2014.

[3] A. Ker and R. Böhme, "Revisiting Weighted Stego-Image Steganalysis," Proc. SPIE: Electronic Imaging, 2008.

[4] S. Li, Y. Shi, and H. Huang, "Text Steganalysis Using Linguistic Features and Machine Learning Techniques," Pattern Recognition Letters, vol. 32, no. 14, pp. 1673–1681, 2011

[5] S. Topkara, M. Topkara, and M. J. Atallah, "Words Are Not Enough: Sentence-Level Linguistic Steganalysis," Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2006.

[6] B. Chen and G. W. Wornell, "Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding," IEEE Transactions on Information Theory, vol. 47, no. 4, pp. 1423–1443, 2001.

[7] H. Farid, "Detection of Hidden Messages Using Higher-Order Statistical Models," Proc. International Conference on Image Processing (ICIP), 2002.

[8] M. Salleh, S. Shanmugam, and M. A. Shanmugam, "A Review of JPEG Steganalysis Techniques," International Journal of Computer Applications, vol. 12, no. 7, pp. 24–35, 2010.

[9] J. Hayes and G. Danezis, "Generating Steganographic Images via Adversarial Training," NeurIPS Workshop on Machine Deception, 2017.

[10] H. Zhang, Y. Guo, L. Zhang, "Deep Learning-Based Image Steganalysis: A Survey," IEEE Access, vol. 7, pp. 172543–172566, 2019.

[11] L. Wu, Z. Ji, and Y. Li, "Detecting Hidden Information in PDFs Using Cross-Layer Structural Analysis," Journal of Digital Forensics, Security and Law, 2021.

[12] R. Bohme, "Advanced Statistical Steganalysis Using Feature Selection," Proc. ACM Multimedia and Security Workshop, 2008.