FAST - NUCES

StegGuard

Detecting Steganography in Images Using Machine Learning

Instructors: Anaum Hamid, Khalid Khan

Section: BCS - 6H

Muhammad Huzaifa (22k4641)

Usman Tanveer (22k4478)

Jaswant Lal (22k4473)

1. Executive Summary

This project aimed to detect steganographic content embedded in digital images using a machine learning-based approach. We developed a pipeline for extracting image features using histograms and trained a Random Forest Classifier for classification. A graphical user interface (GUI) built with Tkinter allows users to upload images and instantly detect hidden content. The model demonstrated exceptional accuracy on training, validation, and test datasets, confirming the effectiveness of the approach.

2. Introduction

Background:

Steganography involves concealing information within media such as images, audio, or video, often making detection challenging. Traditional detection techniques can be limited in effectiveness. Our project uses supervised learning on RGB histograms, leveraging statistical irregularities introduced by data embedding to identify hidden content.

Objectives of the Project:

- Build an end-to-end steganography detection system.
- Extract features using color histograms.
- Train and evaluate a Random Forest classifier.
- Develop a GUI for image upload and prediction.
- Achieve high detection accuracy across various steganographic methods.

3. Project Functionality

Innovations and Modifications:

- Binary classification of images (clean vs. stego).
- Multiple stego types supported (base64, zip, etc.).
- Real-time predictions using a GUI.
- Option to retrain the model using custom datasets.

4. AI Approach and Methodology

AI Techniques Used:

We used a Random Forest Classifier, a robust ensemble learning method known for high accuracy and interpretability. It was trained on histogram-based features from RGB channels of the images.

Algorithm and Heuristic Design:

- Extracted 256-bin histograms from R, G, and B channels using OpenCV.
- Flattened to a 768-length feature vector per image.
- Normalized features to ensure consistent input to the classifier.
- Labels: 0 = clean, 1 = stego.

AI Performance Evaluation:

Training Results:

- Accuracy: 0.99994
- Classification Report:
    - Precision, Recall, F1-Score: All 1.00 for both classes
    - Total Samples: 16,000

Validation Results:

- Accuracy: 0.9985
- Classification Report:
    - Precision, Recall, F1-Score: All 1.00 for both classes
    - Total Samples: 8,000

Testing Results:

- Accuracy: 0.99955
- Classification Report:
    - Precision, Recall, F1-Score: All 1.00 for both classes
    - Total Samples: 20,000

5. Program Workflow

Modified Rules:

Binary classification:

- Class 0 = Clean image
- Class 1 = Stego image

Turn-Based Mechanics:

- The user uploads an image.
- Histogram features are extracted.
- The classifier predicts the image class.

- The result is shown immediately in the GUI.

● Winning Conditions:

The model succeeds when it accurately detects stego content in an image.

6. Implementation and Development

Development Process:

1. Dataset organized into clean and stego categories.
2. Features extracted using histogram analysis.
3. The random forest model was trained and evaluated.
4. Tkinter GUI developed and integrated with the trained model.

Programming Languages and Tools:

- Programming Language: Python
- Libraries: OpenCV, NumPy, Matplotlib, scikit-learn, Tkinter, joblib, PIL
- Tools: Git for version control

Challenges Encountered:

- Encountered UnpicklingError due to .pkl file encoding issues.
- Required consistent feature extraction between GUI and training scripts.
- Ensured high accuracy across different steganographic formats.

7. Team Contributions

- Muhammad Huzaifa (22k4641): Responsible for GUI development and model integration.
- Usman Tanveer (22k4478): Handled feature extraction pipeline and dataset management.
- Jaswant Lal (22k4473): Focused on Random Forest model training, evaluation, and performance testing.

8. Results and Discussion

AI Performance:

- The classifier achieved >99.8% accuracy on all datasets.

- Real-time predictions via GUI with sub-second response time.
- Demonstrated generalization across multiple stego types.
- The model is robust, fast, and effective for binary steganography detection.

9. References

1. Scikit-learn documentation: https://scikit-learn.org/stable/
2. OpenCV documentation: https://docs.opencv.org/
3. Tkinter documentation: https://docs.python.org/3/library/tkinter.html
4. Python Imaging Library (PIL): https://pillow.readthedocs.io/