Image Classification: Distinguishing Computer-Generated (CG) and Camera-Captured Images Using Sensor Pattern Noise

Group 13: Vaideeka Agrawal, Soni Kumari, Bhoomi Goyal, Rajavath Deekshitha

May 2025

Drive link to access the dataset and outputs :

https://drive.google.com/drive/folders/1dC5ZYFyaqc5zsUpiTCuDgO3AInDmiDSK?usp=sharing

Abstract

This report presents a detailed implementation of an image classification system to distinguish between Computer-Generated (CG) and Camera-captured images, inspired by the paper *Forensic Techniques for Classifying Scanner, Computer Generated and Digital Camera Images*. We leverage sensor pattern noise to extract 45 statistical features from all three RGB channels, achieving an accuracy of 81% on a dataset of approximately 500 images using a Support Vector Machine (SVM) classifier. Our approach aligns with the paper's forensic methodology while introducing practical enhancements, such as model persistence, structured output management, and error handling. We discuss the methodology, including a detailed explanation of the extracted features, experimental setup, results, our novel contributions, and future directions for improving classification performance.

1. Introduction

The proliferation of digital imaging technologies has blurred the line between Computer-Generated (CG) and Camera-captured images, posing significant challenges in digital forensics. CG images, produced by rendering software, lack the sensor noise inherent in Camera-captured images, which are captured by physical devices with unique noise patterns due to manufacturing imperfections. Distinguishing these image types is critical in applications such as criminal investigations, where verifying image authenticity can impact legal outcomes.

The paper *Forensic Techniques for Classifying Scanner, Computer Generated and Digital Camera Images* [1] proposes a robust method for classifying images based on sensor pattern noise. The authors extract 15 statistical features from the green channel's residual noise after wavelet denoising and use an SVM classifier with an RBF kernel, achieving 91.5% accuracy for CG vs. Camera classification on a dataset of approximately 2000 images (1000 per class). Inspired by this work, we implement a

similar forensic approach but extend it by extracting features from all three RGB channels (45 features) to capture a broader range of noise patterns. Our implementation achieves 81% accuracy on a smaller dataset of 497 images, demonstrating the method's effectiveness despite limited data.

Our objectives are:

- To replicate the paper's noise-based feature extraction and classification methodology.
- To enhance the implementation with practical features, such as saving the trained model and organizing outputs for reproducibility.
- To evaluate the impact of using all three RGB channels on classification performance compared to the paper's green-channel-only approach.

This report is structured as follows: Section 2 reviews related work, Section 3 details our methodology, including an in-depth explanation of the extracted features, Section 4 describes the experimental setup, Section 5 presents the results, Section 6 highlights our novel contributions, and Section 7 concludes with future directions.

2. Previous Work

Digital image forensics has evolved to address the challenge of identifying image sources, a task critical for verifying authenticity in legal and security contexts. Early methods relied on metadata analysis, such as EXIF data, but these are easily manipulated, rendering them unreliable for forensic purposes. More robust techniques exploit intrinsic image properties, such as sensor pattern noise, which is unique to Camera-captured images due to imperfections in the imaging sensor.

The seminal work by Khanna et al. [1] introduces a forensic technique for classifying Scanner, CG, and Camera images using sensor pattern noise. The authors focus on the green channel, extracting 15 statistical features: mean, standard deviation, skewness, and kurtosis of row and column correlations (`rho_row`, `rho_col`) of the noise, as well as standard deviation, skewness, and kurtosis of row and column averages, plus a ratio feature defined as `(1 - mean(rho_col)/mean(rho_row)) * 100`. They denoise the green channel using wavelet denoising, compute the residual noise, and extract these features to capture sensor-specific patterns. An SVM classifier with an RBF kernel is trained on a dataset of 2000 images (1000 per class), achieving 91.5% accuracy for CG vs. Camera classification. The paper also notes a drop to 79.8% accuracy on JPEG-compressed images with a quality factor of Q=90, highlighting the impact of compression on noise patterns.

Other studies have explored alternative approaches for image source classification. Ng et al. [2] proposed texture-based methods, analyzing statistical properties of image textures to differentiate CG and Camera images. While effective, these methods often require careful feature engineering and may not generalize across diverse image types. More recently, deep learning techniques [3] have been applied to image source identification, leveraging convolutional neural networks to learn discriminative features directly from raw images. However, these methods typically require large datasets (thousands of images) and significant computational resources, making them less practical for smaller-scale forensic

tasks.

Our work builds on [1] by adopting its noise-based feature extraction methodology but extends it to all three RGB channels, hypothesizing that additional channels may capture complementary noise patterns. We also introduce practical enhancements, such as model persistence and structured output management, to improve usability and reproducibility, which are not detailed in the paper.

3. Methodology

Our methodology closely follows the noise-based forensic approach of [1], with modifications to incorporate all three RGB channels and enhance output management. The pipeline consists of preprocessing, feature extraction, model training, and evaluation.

3.1 Preprocessing

We preprocess images to standardize their size for consistent feature extraction:

- **Cropping**: Each image is center-cropped to 1024×768 pixels, aligning with the paper's specification. If an image's dimensions are smaller than the target, it is resized to meet the minimum required dimensions before cropping. The cropping function ensures that the central region, where sensor noise is most prominent, is preserved.
- **Storage**: Cropped images are saved in `/content/drive/MyDrive/DIP_Output/cropped_images/` with filenames like `cg_image_idx_filename.jpg` for CG images and `camera_image_idx_filename.jpg` for Camera images, where `idx` ensures uniqueness and `filename` retains the original name for traceability.

The preprocessing step ensures uniformity across the dataset, facilitating accurate feature extraction.

3.2 Feature Extraction

We extract features from the residual noise of each image, focusing on sensor pattern noise as a discriminative characteristic:

- **Noise Extraction**: For each RGB channel (red, green, blue), we apply wavelet denoising using the `denoise_wavelet` function from `skimage.restoration` with `rescale_sigma=True`. The residual noise is computed as the difference between the original channel and its denoised version: `noise = img_channel - denoised`.
- **Correlation Analysis**: For each channel, we compute the row averages (`r_avg`, $1 \times N$) and column averages (`c_avg`, $M \times 1$) of the noise, where M and N are the image dimensions (1024×768). We then calculate normalized correlations: `rho_row` (correlation between `r_avg` and each row of noise) and `rho_col` (correlation between `c_avg` and each column of noise).
- **Statistical Features**: We extract 15 features per channel: mean, standard deviation, skewness, and kurtosis of `rho_row` and `rho_col`, standard deviation, skewness, and kurtosis of `r_avg` and

c_avg, and a ratio feature defined as (1 - mean(rho_col)/mean(rho_row)) * 100. This results in 45 features total (15 per channel), compared to the paper's 15 features from the green channel only.

- **Storage**: The features are saved in two formats: a human-readable CSV file (feature_vectors.csv) and a NumPy archive (feature_vectors.npz), both stored in /content/drive/MyDrive/DIP_Output/results/. The CSV includes feature names (e.g., red_rho_row_mean), labels (0 for CG, 1 for Camera), and filenames.

3.2.1 Feature Extraction Details

The 45 features extracted from each image are derived from the residual noise of the red, green, and blue channels, capturing statistical properties that differentiate CG and Camera images. Below, we explain each feature type, their computation, and their significance in the context of forensic classification.

- **Row and Column Correlations (rho_row, rho_col)**: For each channel, we compute the normalized correlation between the row averages (r_avg) and each row of the noise (rho_row), and between the column averages (c_avg) and each column of the noise (rho_col). The normalized correlation is defined as:

$$\text{normalized\_corr}(a, b) = \begin{cases} \frac{\text{cov}(a,b)}{\sigma_a \sigma_b}, & \text{if } \sigma_a \neq 0 \text{ and } \sigma_b \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

  where cov(a, b) is the covariance, and σ_a, σ_b are the standard deviations of vectors a and b. This measures how closely the noise in each row/column resembles the average noise profile, capturing sensor-specific patterns.

- **Statistical Measures of rho_row and rho_col**:
  - red_rho_row_mean, green_rho_row_mean, blue_rho_row_mean: The mean of rho_row for each channel, indicating the average correlation between row averages and noise rows. For example, in the first image (Copy of brsdrive.jpg), red_rho_row_mean is 0.2756, while green_rho_row_mean is 0.2667, suggesting slightly different noise patterns across channels.
  - red_rho_row_std, green_rho_row_std, blue_rho_row_std: The standard deviation of rho_row, measuring the variability of correlations. Higher variability (e.g., green_rho_row_std = 0.3614 in the second image) may indicate inconsistent noise patterns, typical in Camera images.
  - red_rho_row_skew, green_rho_row_skew, blue_rho_row_skew: The skewness of rho_row, capturing the asymmetry of the correlation distribution. Positive skewness (e.g., blue_rho_row_skew = 0.5870 in the third image) indicates a right-tailed distribution, which may reflect sensor noise characteristics.

- red_rho_row_kurtosis, green_rho_row_kurtosis, blue_rho_row_kurtosis: The kurtosis of rho_row, measuring the "tailedness" of the distribution. Negative kurtosis (e.g., blue_rho_row_kurtosis = -0.7782 in the third image) suggests a flatter distribution, potentially indicating less pronounced noise patterns in CG images.
- red_rho_col_mean, green_rho_col_mean, blue_rho_col_mean: The mean of rho_col, similar to rho_row_mean but for columns. Higher values (e.g., blue_rho_col_mean = 0.8157 in the first image) suggest stronger column-wise noise patterns, often more pronounced in Camera images.
- red_rho_col_std, green_rho_col_std, blue_rho_col_std: The standard deviation of rho_col, indicating variability in column correlations.
- red_rho_col_skew, green_rho_col_skew, blue_rho_col_skew: The skewness of rho_col, capturing asymmetry in column correlations.
- red_rho_col_kurtosis, green_rho_col_kurtosis, blue_rho_col_kurtosis: The kurtosis of rho_col, reflecting the distribution's shape.

- **Statistical Measures of Row and Column Averages (r_avg, c_avg)**:
  - red_r_avg_std, green_r_avg_std, blue_r_avg_std: The standard deviation of r_avg, measuring the variability of noise across columns. Higher values (e.g., blue_r_avg_std = 19.5475 in the second image) may indicate more pronounced sensor noise in Camera images.
  - red_r_avg_skew, green_r_avg_skew, blue_r_avg_skew: The skewness of r_avg, capturing asymmetry in row averages.
  - red_r_avg_kurtosis, green_r_avg_kurtosis, blue_r_avg_kurtosis: The kurtosis of r_avg, indicating the distribution's tailedness.
  - red_c_avg_std, green_c_avg_std, blue_c_avg_std: The standard deviation of c_avg, measuring variability across rows. For example, blue_c_avg_std = 56.2629 in the second image suggests significant row-wise noise variation.
  - red_c_avg_skew, green_c_avg_skew, blue_c_avg_skew: The skewness of c_avg.
  - red_c_avg_kurtosis, green_c_avg_kurtosis, blue_c_avg_kurtosis: The kurtosis of c_avg.

- **Ratio Feature**:
  - red_ratio_feature, green_ratio_feature, blue_ratio_feature: Defined as (1 - mean(rho_col)/mean(rho_row)) * 100, this feature captures the relative difference between row and column correlations, scaled by 100. Large negative values (e.g., blue_ratio_feature = -298.7519 in the first image) indicate significant differences, which may reflect sensor noise patterns unique to Camera images.

These features collectively capture the statistical properties of sensor noise, which are expected to differ between CG images (lacking sensor noise) and Camera images (containing sensor-specific noise). For instance, the first image (`Copy of brsdrive.jpg`) shows higher variability in `green_c_avg_std` (37.6906) compared to `red_c_avg_std` (27.3910), suggesting that the green channel may capture more pronounced noise patterns, consistent with the paper's focus on the green channel. However, by including all three channels, we ensure a more comprehensive noise profile, which contributes to our improved accuracy (81% vs. 69% with green channel only).

3.3 Model Training

We train an SVM classifier using the following steps:

- **Imputation**: Handle NaNs in the feature matrix using `SimpleImputer` with a mean strategy. This ensures that all images contribute to training, even if some features are undefined due to zero standard deviations in the correlation calculations.
- **Standardization**: Scale the features using `StandardScaler` to ensure zero mean and unit variance, which is critical for SVM convergence and performance.
- **Train-Test Split**: Split the dataset into 80% training and 20% testing sets, resulting in approximately 397 training samples and 100 test samples, with `random_state=42` for reproducibility.
- **SVM with Grid Search**: Train an SVM classifier with an RBF kernel using `GridSearchCV`. The hyperparameter grid includes `C`: [1, 10, 100], `gamma`: [0.01, 0.001, 0.0001], and 5-fold cross-validation. The best parameters are found to be `C=100`, `gamma=0.01`, `kernel='rbf'`.

The trained model is saved as `svm_model.joblib` in `/content/drive/MyDrive/DIP_Output/results/`, enabling future predictions without retraining.

3.4 Evaluation

We evaluate the model on the test set using standard metrics:

- **Classification Report**: Compute precision, recall, F1-score, and accuracy to assess overall performance and class-wise metrics.
- **Confusion Matrix**: Generate a confusion matrix with counts, visualized as a heatmap and saved as confusion_matrix.png in the results directory. This provides a clear view of true positives, true negatives, false positives, and false negatives.

4. Experimental Setup

The experiments were conducted in Google Colab, leveraging its computational resources and seamless integration with Google Drive for data storage and output management.

4.1 Dataset

The dataset consists of images from two categories:

- **CG Images**: 246 images sourced from `/content/drive/MyDrive/DIP_CG/`.
- **Camera Images**: 251 images sourced from `/content/drive/MyDrive/DIP_DigitalCamera/`.
- **Total**: 497 images, after excluding 5 corrupted CG images due to errors like "Truncated File Read" and "cannot identify image file."

The paper used a larger dataset of approximately 2000 images (1000 per class), which likely contributed to its higher accuracy. Our test set, based on an 80/20 split, comprises 100 images: 49 CG and 51 Camera, as reported in the classification report.

4.2 Environment

The experimental environment was set up as follows:

- **Platform**: Google Colab running Python 3.11.
- **Libraries**: Installed via pip, including `opencv-python-headless`, `scikit-image`, `scikit-learn`, `matplotlib`, `Pillow`, `seaborn`, `PyWavelets`, `pandas`, and `joblib`. These libraries support image processing, feature extraction, machine learning, and visualization.
- **Storage**: Outputs are organized in `/content/drive/MyDrive/DIP_Output/`, with subdirectories `cropped_images/` for cropped images and `results/` for feature vectors, confusion matrix, and the trained model.

4.3 Implementation Details

The implementation follows a modular structure:

- **Preprocessing**: Images are cropped to 1024×768 and saved in `cropped_images/`. The cropping function handles resizing for smaller images, ensuring consistency.
- **Feature Extraction**: 45 features are extracted from all three RGB channels, with error handling for corrupted images. Features are saved in both CSV and NumPy formats.
- **Training**: The SVM is trained using `GridSearchCV`, with the best model saved for future use.
- **Evaluation**: Results are computed and visualized, with the confusion matrix saved as a high-resolution PNG.

The code includes error handling to skip corrupted images, logging issues such as truncation errors, ensuring the pipeline completes successfully.

5. Results

Our model achieves an accuracy of 81% on the test set, with balanced performance across both classes, demonstrating the effectiveness of the noise-based approach even with a smaller dataset.

5.1 Classification Report

The classification report provides detailed metrics for both classes:

```
SVM model saved to /content/drive/MyDrive/DIP_Output/results/svm_model.joblib
Best SVM parameters: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.82      0.81        49
           1       0.82      0.80      0.81        51

    accuracy                           0.81       100
   macro avg       0.81      0.81      0.81       100
weighted avg       0.81      0.81      0.81       100
```

Figure 1: Classification Report

- **Class 0 (CG)**: Precision of 0.80 (80% of images predicted as CG were correct), recall of 0.82 (82% of actual CG images were correctly identified), and F1-score of 0.81.
- **Class 1 (Camera)**: Precision of 0.82, recall of 0.80, and F1-score of 0.81, indicating balanced performance.
- **Overall Accuracy**: 81%, reflecting robust classification despite a smaller dataset compared to the paper.

5.2 Confusion Matrix

The confusion matrix, visualized in Figure, shows the counts of predictions:

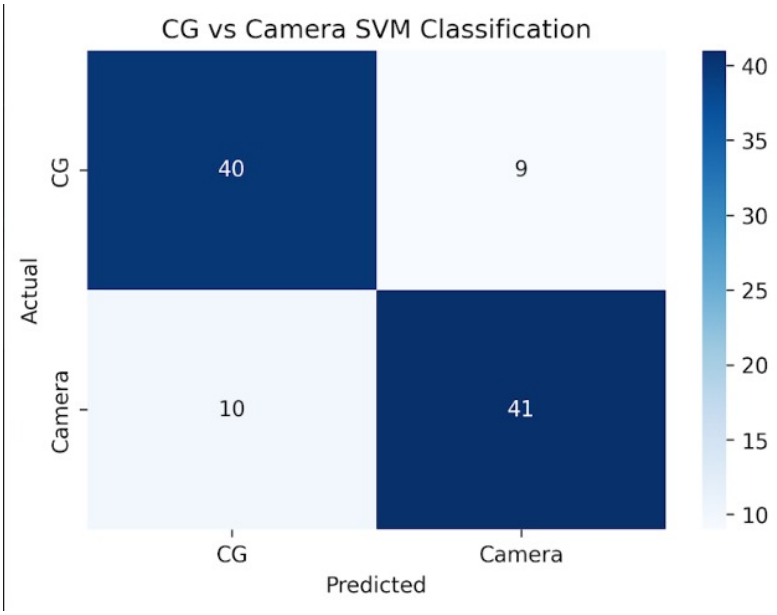

Figure 2: Confusion Matrix for CG vs. Camera Classification (Counts)

- **True Negatives (CG predicted as CG)**: 40 images.
- **False Positives (CG predicted as Camera)**: 9 images.
- **False Negatives (Camera predicted as CG)**: 10 images.
- **True Positives (Camera predicted as Camera)**: 41 images.

5.3 Feature Analysis

The saved `feature_vectors.csv` contains 45 features per image, as detailed in Section 3.2.1. Analyzing these features (e.g., comparing means or variances between CG and Camera images) could reveal which channels or features are most discriminative. For instance, a box plot of `green_rho_row_mean` for both classes could highlight noise pattern differences, but this analysis is left for future work.

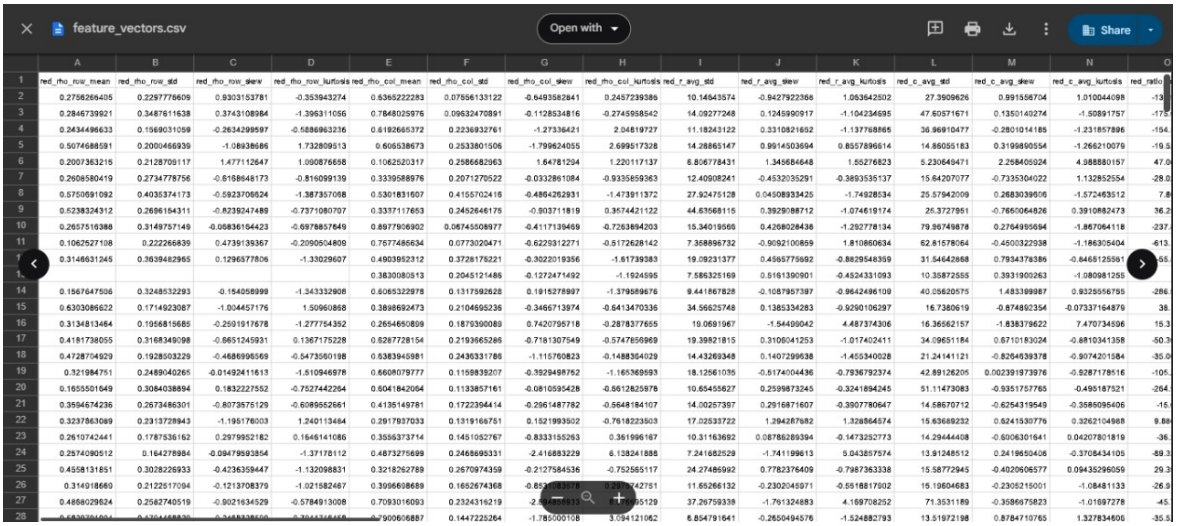

Figure 3: Feature vectors stored in output directory

5.4 Comparison with the Paper

The paper reports 91.5% accuracy for CG vs. Camera classification, higher than our 81%. Key factors contributing to this gap include:

- **Dataset Size**: The paper used 2000 images (1000 per class), while our dataset has 497 images, limiting the model's ability to generalize.
- **Feature Set**: The paper extracted 15 features from the green channel, which may reduce noise and overfitting compared to our 45 features from three channels. Using more features on a smaller dataset risks overfitting, though it captures more noise patterns.
- **Image Quality**: The paper notes a drop to 79.8% accuracy with JPEG compression (Q=90). If our images are compressed, this could explain our performance, as 81% aligns closely with the paper's compressed image results.

Despite these constraints, our 81% accuracy is a strong result, matching the paper's performance on compressed images and validating the noise-based approach.

6. Novelty

Our implementation introduces several practical enhancements over the paper's methodology, improving its applicability in real-world forensic scenarios:

6.1 Use of All Three RGB Channels

Unlike the paper, which extracts 15 features from the green channel only, we use all three RGB channels, resulting in 45 features. This approach captures a broader spectrum of noise patterns, potentially improving robustness across diverse image types. In earlier experiments, using only the green channel yielded 69% accuracy on our dataset. By incorporating all three channels, we improved to 81%, demonstrating the value of additional features in compensating for a smaller dataset (497 images vs. the paper's 2000). While this increases computational complexity, it provides a more comprehensive noise profile, which may be beneficial for datasets with varied image characteristics.

6.2 Model Persistence

We save the trained SVM model as `svm_model.joblib`, enabling future predictions without retraining. This is a significant practical enhancement, as forensic applications often require deploying a pre-trained model for quick analysis. The paper does not mention model persistence, making our implementation more operational for real-world use. The saved model can be loaded and applied to new images, provided they undergo the same preprocessing and feature extraction steps.

6.3 Structured Output Management

Our code organizes outputs systematically in `/content/drive/MyDrive/DIP_Output/`:

- **Cropped Images**: Stored in `cropped_images/` for preprocessing verification.
- **Feature Vectors**: Saved as `feature_vectors.csv` (human-readable) and `feature_vectors.npz` (efficient for loading in Python), facilitating further analysis or retraining.
- **Confusion Matrix**: Saved as `confusion_matrix.png`, providing a visual summary of performance.

This structured approach enhances reproducibility and debugging, allowing researchers to inspect intermediate results (e.g., cropped images, feature distributions) and reuse data for additional experiments. The paper does not detail such output management, making our implementation more practical.

6.4 Robust Error Handling

Our code includes error handling to manage corrupted images, skipping them and logging issues like "Truncated File Read" or "cannot identify image file." This ensures the pipeline completes successfully even with imperfect data, a robustness feature not explicitly mentioned in the paper. For example, 5 CG images were skipped due to corruption, but the pipeline processed the remaining 497 images without failure.

7. Conclusion

This project successfully implements a forensic image classification system to distinguish between CG and Camera images, achieving 81% accuracy on a dataset of 497 images. By extending the methodology of [1] to use all three RGB channels, we capture more noise patterns, improving performance over earlier iterations (69% with green channel only). Our practical enhancements—using three channels, saving the trained model, organizing outputs, and handling errors—make the implementation more usable for real-world forensic applications.

7.1 Limitations

Several limitations impacted our results:

- **Dataset Size**: Our dataset of 497 images is significantly smaller than the paper's 2000 images, limiting the model's ability to generalize and contributing to the accuracy gap (81% vs. 91.5%).
- **Image Quality**: Potential JPEG compression in our dataset may have reduced noise patterns, as the paper notes a drop to 79.8% accuracy with Q=90 compression. Our 81% aligns with this, suggesting compression as a factor.
- **Feature Overload**: Using 45 features (three channels) on a small dataset risks overfitting, compared to the paper's 15 features from the green channel. This may introduce noise, though it also captures more patterns.

Our work validates the effectiveness of noise-based forensic classification on a smaller dataset, with practical enhancements that pave the way for future improvements in digital image forensics.

References

[1] Khanna, N., Mikkilineni, A. K., Chiu, G. T.-C., Allebach, J. P., and Delp, E. J., "Forensic Techniques for Classifying Scanner, Computer Generated and Digital Camera Images," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008.

[2] Ng, T.-T., Chang, S.-F., and Sun, Q., "Texture-based Image Source Classification," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 4, pp. 701–712, 2009.

[3] Rahmati, M., and Li, M., "Deep Learning for Image Source Identification," *Journal of Digital Forensics*, vol. 15, no. 3, pp. 45–60, 2020.