Fake Colorized Image Detection with Channel Difference Maps documentation

---

AI-Generated Content Disclaimer

This document has been generated using GPT-based AI. While efforts have been made to ensure accuracy, the content may contain irrelevant, outdated, or imprecise information. Users are advised to critically evaluate the information provided and verify important details independently. Any inconsistencies or errors should be disregarded.

I. Introduction

The proliferation of sophisticated image editing tools, driven by advancements in deep learning, has led to a surge in artificially colorized images. While offering creative possibilities and aiding in historical preservation, this technology also poses a threat to the authenticity of digital media. 1 The ability to accurately distinguish between naturally colored and artificially colorized images is crucial for maintaining trust in digital content across various domains, including forensics, journalism, and historical archiving. 1 Traditional image manipulation detection techniques, which often rely on identifying noise patterns or compression artifacts, are becoming less effective against the realistic outputs of deep learning-based colorization methods. 2

This paper introduces a novel approach that leverages Channel Difference Maps (CDMs) and a dual-network architecture to address the challenges of fake colorized image detection. By analyzing the subtle inconsistencies in color channel relationships, the proposed method offers a robust solution for identifying artificially colorized images.

II. Related Work

A. Image Forgery Detection

Image forgery detection is a well-established field with a range of techniques. These techniques can be broadly categorized into pixel-based, format-based, and physics-based methods. Pixel-based methods examine individual pixels and their relationships, format-based methods analyze file formats and compression artifacts, and physics-based methods study illumination and other physical properties. Deep learning has recently emerged as a powerful tool in image forgery detection, demonstrating significant potential in identifying various types of image manipulations, such as splicing, copy-move forgeries, and object removal.

B. Deep Learning for Colorization Detection

Several studies have explored the use of deep learning for detecting artificially colorized images. These methods typically employ Convolutional Neural Networks (CNNs) to extract relevant features from images and perform classification. However, many existing approaches treat this as a general binary classification problem, without specifically addressing the unique color inconsistencies that are characteristic of artificially colorized images.

C. Channel Analysis in Image Processing

Channel analysis is a fundamental technique in image processing, used in applications like image enhancement, segmentation, and compression. Analyzing the relationships between different color channels can reveal valuable information about the image content and any alterations it may have undergone. This research builds upon this principle by introducing Channel Difference Maps (CDMs) as a specialized feature representation for the task of colorization detection.

III. Proposed Method

The proposed method uses Channel Difference Maps (CDMs) to extract distinctive features from images and a dual-network architecture to effectively detect artificially colorized images.

A. Channel Difference Maps (CDMs)

Channel Difference Maps (CDMs) are designed to capture the relationships between the red (R), green (G), and blue (B) color channels of an image. The rationale behind using CDMs is that artificially colorized images often exhibit inconsistencies in the statistical relationships between color channels compared to naturally colored images.

The CDM is computed by calculating the difference between each pair of color channels:

1. R-G: The difference between the red and green channels.

2. G-B: The difference between the green and blue channels.

3. R-B: The difference between the red and blue channels.

These three difference maps are then combined into a three-channel representation, where each channel represents one of the color differences.

$$CDM = [R-G, G-B, R-B]$$

This representation highlights the variations in color distribution and correlation, providing a more sensitive measure for distinguishing between real and fake colorized images.

B. Dual-Network Architecture

The core of the proposed method is a dual-network architecture, consisting of a regeneration network (autoencoder) and a detection network. A key feature of this architecture is the sharing of encoder weights between the two networks. This weight sharing allows the

detection network to benefit from the feature learning capabilities of the regeneration network.

Regeneration Network (Autoencoder)

- The regeneration network is designed as an autoencoder, which learns to reconstruct the original image from its Channel Difference Map representation.
- It follows a U-Net-like architecture, incorporating skip connections between the encoder and decoder paths. This architecture is well-suited for image reconstruction tasks, as skip connections help to preserve fine-grained details

The regeneration network consists of:

- Three parallel encoder paths: Each encoder path processes one channel of the CDM (R-G, G-B, or R-B) and consists of multiple convolutional blocks.
- A shared dense feature extraction block: This block processes the concatenated features from the encoder paths to capture high-level representations
- A decoder path with skip connections: The decoder reconstructs the image from the extracted features, using skip connections to incorporate information from the encoder paths at different resolutions.
- The encoder paths use convolutional layers with increasing filter sizes, batch normalization, LeakyReLU activation, and max pooling to extract features at different scales.

Detection Network:

The detection network shares the same encoder architecture as the regeneration network. However, instead of a decoder, it uses a classification head to determine whether an input image is real or fake.

The detection network consists of:

- Three parallel encoder paths: Identical to those in the regeneration network.
- Residual blocks for feature refinement: These blocks further process the extracted features to enhance their discriminative power.
- Convolutional layers and fully connected layers: These layers form the classification head, which outputs the probability of the input image being real or fake.
- The key idea is to transfer the weights of the encoder paths from the trained regeneration network to the detection network. This transfer learning technique allows the detection network to leverage the features learned by the regeneration network, which are specifically tailored for representing Channel Difference Maps.

C. Training Strategy

The training process is divided into two distinct phases to optimize the performance of both networks:

Phase 1: Regeneration Network Training

- The regeneration network (autoencoder) is trained to reconstruct images from their Channel Difference Maps.
- The network is trained using the Mean Squared Error (MSE) loss function, which measures the difference between the original and reconstructed images.
- This phase enables the encoder to learn effective representations of CDMs for image reconstruction.

Phase 2: Detection Network Training

- The encoder weights learned during the regeneration network training are transferred to the detection network.
- The detection network is then trained for binary classification, with the goal of distinguishing between real and fake images.
- The detection network is also trained using the Mean Squared Error (MSE) loss function, although other classification loss functions could be used.
- This two-phase approach allows the detection network to benefit from pre-trained feature extractors that are specifically designed to capture the characteristics of Channel Difference Maps.

IV. Implementation Details

A. Dataset Preparation

The performance of any deep learning model heavily relies on the quality and preparation of the dataset. In this work, the dataset is organized into two main categories: real (naturally colored) images and fake (artificially colorized) images.

Dataset Structure

- The dataset is structured with real and fake images stored in separate directories. This organization simplifies data loading and processing.

Preprocessing Steps

- To ensure consistency and optimize training, the images undergo several preprocessing steps:
  - Color Space Conversion: Images are loaded and converted from the BGR (Blue-Green-Red) color space to the RGB (Red-Green-Blue) color space. This is necessary because some image processing libraries use BGR as the default color space, while the model is designed to work with RGB.
  - Resizing: Images are resized to a fixed size. The default size is 256x256 pixels, but this can be adjusted depending on the specific requirements and computational resources. Resizing ensures that all input images have the same dimensions, which is essential for batch processing in neural networks.
  - Normalization: Pixel values are normalized to the range [-1, 1]. This is done by dividing the pixel values by 127.5 and subtracting 1. Normalization helps to improve the training stability and speed by preventing large pixel values from dominating the learning process.

Data Generators

- For efficient training, data generators are created using libraries like Keras.
    - These generators load images in batches, preprocess them on-the-fly, and feed them to the model during training.
    - Data generators also handle shuffling of the data to reduce bias and improve generalization.
    - The dataset is typically split into training and validation sets. The training set is used to train the model, while the validation set is used to monitor its performance during training and prevent overfitting.

The networks are implemented using a deep learning framework like TensorFlow with Keras. This section outlines the key components and architectural choices.

1. Custom Lambda Layer for CDM Computation

- A custom layer is defined to compute the Channel Difference Maps within the network. This layer takes an RGB image tensor as input and outputs the corresponding CDM tensor.
- Using a custom layer allows the CDM computation to be seamlessly integrated into the network architecture and benefit from automatic differentiation during training.

2. Encoder Path Architecture

- Each encoder path (one for each CDM channel) consists of a series of convolutional blocks.
- Each convolutional block typically includes:
    - Convolutional Layer: Extracts local features from the input.
    - Batch Normalization: Normalizes the activations to improve training stability.
    - LeakyReLU Activation: Introduces non-linearity.
    - Max Pooling Layer: Downsamples the feature maps, reducing spatial dimensions and increasing the receptive field.
- The number of filters in the convolutional layers increases progressively in each block, allowing the network to learn features at different levels of abstraction.

3. Scale Block

- The scale block is used in the dense feature extraction part of the regeneration network.
- It typically consists of:
    - Convolutional Layer: Processes the input features.
    - Batch Normalization: Normalizes the activations.
    - LeakyReLU Activation: Introduces non-linearity.
- Multiple scale blocks are stacked to further refine the extracted features.

4. Residual Block

- Residual blocks are used in the detection network to refine the features extracted by the encoder.
- A residual block includes:
    - Convolutional Layers: Process the input features.
    - Batch Normalization: Normalizes the activations.
    - LeakyReLU Activation: Introduces non-linearity.

    - Skip Connection: Adds the original input to the output of the convolutional layers. This helps to alleviate the vanishing gradient problem and improve training.
- If the number of input and output channels in the residual block is different, a 1x1 convolutional layer is used in the skip connection to project the input to the correct dimensions.

5. Regeneration Network Architecture

- The regeneration network (autoencoder) is built using the encoder paths, scale blocks, and decoder paths.
- The input to the network is an RGB image.
- The CDM is computed using the custom Lambda layer.
- The CDM is split into its three channels (R-G, G-B, R-B), and each channel is fed into a separate encoder path.
- The outputs of the encoder paths are concatenated and fed into a series of scale blocks to extract dense features.
- The decoder reconstructs the image from the dense features, using upsampling layers and skip connections to incorporate information from the encoder paths.
- The output of the regeneration network is the reconstructed RGB image.

6. Detection Network Architecture

- The detection network shares the same encoder architecture as the regeneration network.
- The input to the network is an RGB image.
- The CDM is computed using the custom Lambda layer.
- The CDM is split into its three channels, and each channel is fed into a separate encoder path.
- The outputs of the encoder paths are concatenated and fed into a series of residual blocks to refine the extracted features.
- The refined features are then fed into a classification head, which typically consists of:
    - Convolutional Layers: Further process the features.
    - Flatten Layer: Converts the feature maps into a 1D vector.
    - Dense Layers (Fully Connected Layers): Perform the final classification.
    - Dropout Layers: Regularize the network to prevent overfitting.
    - Softmax Activation: Outputs the probability of the input image being real or fake.
- The output of the detection network is a probability distribution over the two classes (real and fake).

C. Training Process

The training process is crucial for optimizing the network parameters and achieving high performance.

1. Regeneration Network Training

- The regeneration network is trained to minimize the Mean Squared Error (MSE) between the original and reconstructed images.
- The MSE loss function is defined as:

- MSE = $(1/n) * \Sigma (y_i - \hat{y}_i)^2$
    - where:
    - n is the number of pixels in the image.
    - y_i is the pixel value of the original image.
    - $\hat{y}_i$ is the pixel value of the reconstructed image.
- The network is trained using an optimization algorithm such as Stochastic Gradient Descent (SGD) or Adam.
- Early stopping is used to prevent overfitting. This technique monitors the performance of the network on a validation set and stops training when the performance starts to degrade.

2. Detection Network Training

- After training the regeneration network, the encoder weights are transferred to the detection network.
- The detection network is then trained to minimize the MSE loss for binary classification. While MSE can be used, Cross-Entropy loss is more common and often more effective for classification tasks.
- For binary classification, the Cross-Entropy loss is defined as:
- Cross-Entropy = - [y * log(p) + (1 - y) * log(1 - p)]
- where:
    - y is the true label (0 for real, 1 for fake).
    - p is the predicted probability of the image being fake.
- The network is trained using an optimization algorithm such as SGD or Adam.
- Early stopping is also used during the detection network training to prevent overfitting.

3. Transfer Learning

- Transfer learning is a key component of the proposed method.
- The encoder weights learned during the regeneration network training are transferred to the detection network.
- This allows the detection network to benefit from the feature extraction capabilities of the regeneration network, which has been specifically trained to represent Channel Difference Maps.
- Transfer learning can significantly speed up training and improve the performance of the detection network.

D. Evaluation Metrics

To quantitatively assess the performance of the fake colorized image detection system, several evaluation metrics are used. These metrics provide insights into different aspects of the system's accuracy and reliability.

1. Accuracy

- Accuracy is the most straightforward metric, representing the overall percentage of correctly classified images.
- It is calculated as:
- Accuracy = (Number of Correctly Classified Images) / (Total Number of Images)
- While easy to understand, accuracy can be misleading if the dataset is imbalanced (i.e., if there are significantly more real or fake images).

2. **Precision**

- Precision measures the proportion of images classified as "fake" that are actually fake.
- It is calculated as:
- Precision = (True Positives) / (True Positives + False Positives)
- High precision indicates that the system has a low rate of false positives (i.e., it rarely incorrectly identifies real images as fake).

3. **Recall (Sensitivity)**

- Recall measures the proportion of actual "fake" images that are correctly classified as fake.
- It is calculated as:
- Recall = (True Positives) / (True Positives + False Negatives)
- High recall indicates that the system has a low rate of false negatives (i.e., it rarely fails to identify fake images).

4. **F1 Score**

- The F1 score is the harmonic mean of precision and recall.
- It provides a balanced measure of the system's performance, especially when precision and recall are uneven.

5. It is calculated as:

6. F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

- ROC AUC (Area Under the Receiver Operating Characteristic Curve)
    - The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds.
- The AUC represents the overall ability of the system to distinguish between real and fake images.
- An AUC of 1 indicates perfect classification, while an AUC of 0.5 indicates random guessing.

1. **HTER (Half Total Error Rate)**

- The Half Total Error Rate (HTER) is a metric commonly used in biometric authentication and can be adapted for this task.
- It is calculated as the average of the False Positive Rate (FPR) and the False Negative Rate (FNR):
- HTER = (FPR + FNR) / 2
- The FPR is the proportion of real images incorrectly classified as fake.
- The FNR is the proportion of fake images incorrectly classified as real.
- HTER provides a balanced measure of both types of errors.

V. Experimental Results

This section describes the experimental setup and results obtained from evaluating the proposed fake colorized image detection system.

A. Experimental Setup

1. **Dataset**

- A dataset comprising both naturally colored (real) and artificially colorized (fake) images is used for training and evaluation.
- The dataset is divided into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune hyperparameters and prevent overfitting, and the test set is used to evaluate the final performance of the trained model.
- The dataset should contain a diverse range of images to ensure that the model generalizes well to different image content and colorization styles.

1. Implementation Details

- The system is implemented using Python and the TensorFlow/Keras deep learning framework.
- The training process is performed on a GPU to accelerate computation.
- Hyperparameters such as batch size, learning rate, and number of epochs are carefully tuned using the validation set.

2. Baseline Methods

- The performance of the proposed method is compared with several baseline methods for fake colorized image detection.
- These baseline methods may include traditional image processing techniques, existing deep learning-based approaches, or state-of-the-art methods reported in the literature.

B. Results and Analysis

1. Quantitative Results

- The performance of the proposed method and the baseline methods is evaluated using the metrics described in Section IV.D.
- The results are presented in tables and graphs, showing the accuracy, precision, recall, F1 score, ROC AUC, and HTER for each method.
- Statistical significance tests are performed to determine whether the differences in performance between the methods are statistically significant.

2. Qualitative Results

- In addition to quantitative metrics, qualitative results are also presented.
- This includes examples of correctly and incorrectly classified images, as well as visualizations of the Channel Difference Maps and feature maps learned by the network.
- Qualitative analysis provides insights into the strengths and weaknesses of the proposed method and helps to understand how it works.

3. Ablation Studies

- Ablation studies are performed to analyze the contribution of different components of the proposed method.
- For example, the impact of using Channel Difference Maps, the dual-network architecture, or transfer learning can be investigated.
- Ablation studies help to understand the importance of each component and to identify potential areas for improvement.

4. Discussion

- The results are discussed in detail, analyzing the performance of the proposed method in comparison to the baseline methods.
- The strengths and weaknesses of the proposed method are highlighted.
- The implications of the results for the field of fake colorized image detection are discussed.
- Future research directions are suggested.

VI. System Architecture and Data Flow

A. System Architecture

The fake colorized image detection system is designed with a modular architecture, promoting flexibility, maintainability, and extensibility. The system comprises several key modules, each responsible for a specific function:

1. Data Utilities Module

This module handles all aspects of data management, including loading, preprocessing, and batch generation.

- It provides functions to read images from directories, convert color spaces, resize images, normalize pixel values, and create data generators for training and validation.
- The data generators are essential for efficient training, as they load and preprocess images in batches, feeding them to the model during the training process.

2. CDM Utilities Module

- This module is specifically designed for computing and visualizing Channel Difference Maps (CDMs).
- It includes functions to calculate the CDM from an RGB image and to visualize the individual R-G, G-B, and R-B channels.
- The visualization tools are valuable for understanding the characteristics of CDMs and for debugging the system.

3. Models Module

- This module defines the architectures of the regeneration network (autoencoder) and the detection network.
- It contains functions to build the different components of the networks, such as encoder paths, scale blocks, residual blocks, and classification heads.
- The models module encapsulates the network structures, making it easy to modify or experiment with different architectures.

4. Training Module

- This module implements the training procedures for both the regeneration and detection networks.
- It provides functions to compile the models, train them using specified loss functions and optimizers, and transfer encoder weights from the regeneration network to the detection network.
- The training module also incorporates techniques like early stopping to prevent overfitting and improve training efficiency.

5. Evaluation Module

- This module provides functions for evaluating the performance of the trained detection model.
- It calculates various evaluation metrics, such as accuracy, precision, recall, F1 score, ROC AUC, and HTER, to quantitatively assess the model's effectiveness.
- The evaluation module helps to provide a comprehensive understanding of the model's strengths and weaknesses.

6. Visualization Module

- This module offers a suite of tools for visualizing the training process and the model's results.
- It includes functions to plot training history curves (loss and accuracy), generate confusion matrices, and plot ROC curves.
- The visualization module aids in understanding the model's learning behavior and interpreting its performance.

B. Data Flow

- The data flow within the system describes the sequence of operations performed on the input data to achieve fake colorized image detection. The typical data flow is as follows:

1. Data Loading and Preprocessing

- The process begins with loading the image data from the dataset.
- Images are then preprocessed to ensure consistency and optimize the subsequent processing steps.
- Preprocessing may involve color space conversion (e.g., BGR to RGB), resizing to a standard dimension, and normalizing pixel values.

2. Channel Difference Map (CDM) Generation

- The preprocessed RGB images are then used to generate Channel Difference Maps (CDMs).
- This involves computing the differences between the red, green, and blue color channels to extract features that highlight color inconsistencies.

3. Regeneration Network Processing (Encoder Feature Learning)

- The generated CDMs are fed into the regeneration network (autoencoder).
- The encoder part of the regeneration network learns to extract meaningful feature representations from the CDMs.
- The decoder part of the regeneration network attempts to reconstruct the original image from these feature representations.

4. Encoder Weight Transfer

- After the regeneration network is trained, the learned weights of its encoder are transferred to the encoder of the detection network.
- This transfer learning technique allows the detection network to leverage the feature extraction capabilities developed during the regeneration phase.

5. Detection Network Processing (Classification)

- The input images (or their CDMs) are fed into the detection network.
- The encoder extracts features, and the classification head processes these features to determine whether the image is real or fake.

6. Performance Evaluation

- The output of the detection network (classification results) is compared with the ground truth labels.
- Evaluation metrics are calculated to assess the accuracy and effectiveness of the detection process.

7. Result Visualization

- The training process and the evaluation results are visualized to gain insights into the system's behavior and performance.
- Visualization tools, such as plots and confusion matrices, are used to present the information in a clear and understandable manner.

VII. Algorithm Analysis

- This section provides a detailed analysis of the key algorithms used in the fake colorized image detection system.

A. Channel Difference Map (CDM) Generation Algorithm

- The CDM generation algorithm is responsible for extracting color difference information from an RGB image.
- Input: An RGB image represented as a 3D tensor of shape (height, width, 3), where each channel represents the red, green, and blue color values.
- Output: A CDM represented as a 3D tensor of the same shape (height, width, 3), where each channel represents the R-G, G-B, and R-B color differences.

Algorithm Steps:

1. Extract Color Channels:

- Separate the input RGB image into its individual red (R), green (G), and blue (B) channels.

2. Compute Color Differences:

- Calculate the difference between the red and green channels: R_G = R - G
- Calculate the difference between the green and blue channels: G_B = G - B
- oCalculate the difference between the red and blue channels: R_B = R - B

3. Concatenate Difference Maps:

- Concatenate the resulting R_G, G_B, and R_B difference maps along the channel dimension to form the final CDM.

4. Encoder Path Algorithm

5. The encoder path algorithm is used to extract hierarchical features from the input CDM channels.

C. Regeneration Network Training Algorithm

- The regeneration network training algorithm trains the autoencoder to reconstruct images from their CDM representations.
    - Input: A set of training images and their corresponding CDMs.
    - Output: A trained regeneration network model.

Algorithm Steps:

1. Initialize Network:

Build the regeneration network architecture.

Define the Mean Squared Error (MSE) loss function.

Select an optimization algorithm (e.g., SGD, Adam).

2 Iterate Through Epochs:

- For each epoch:
- Iterate Through Training Batches:
    - For each batch of training images and CDMs:
    - Forward Pass: Pass the CDM through the network to generate a reconstructed image.
    - Compute Loss: Calculate the MSE loss between the original image and the reconstructed image.
    - Backward Pass: Compute the gradients of the loss with respect to the network parameters.
    - Update Parameters: Update the network parameters using the chosen optimization algorithm and the computed gradients.
    - Evaluate on Validation Set (Optional):
    - Calculate the loss on a validation set to monitor performance and prevent overfitting.
    - Check for Early Stopping:
    - If the validation loss has not improved for a certain number of epochs (patience), stop training.

3. Output Trained Model:

- Return the trained regeneration network model.

D. Detection Network Training Algorithm

- The detection network training algorithm trains the classifier to distinguish between real and fake colorized images.
    - Input: A pre-trained regeneration network model, a set of training images, and their corresponding labels (real or fake).
    - Output: A trained detection network model.

Algorithm Steps:

1. Initialize Network:

- Build the detection network architecture.
- Transfer the encoder weights from the pre-trained regeneration network to the detection network.
- Define a loss function suitable for binary classification (e.g., Cross-Entropy loss, MSE loss).
- Select an optimization algorithm (e.g., SGD, Adam).

2. **Iterate Through Epochs:**

- For each epoch:
    - Iterate Through Training Batches:
    - For each batch of training images and labels:
    - Forward Pass: Pass the input image through the network to generate a probability of it being fake.
    - Compute Loss: Calculate the classification loss between the predicted probabilities and the true labels.
    - Backward Pass: Compute the gradients of the loss with respect to the network parameters.
    - Update Parameters: Update the network parameters using the chosen optimization algorithm and the computed gradients.
    - Evaluate on Validation Set (Optional):
    - Calculate the classification accuracy and other metrics on a validation set.
    - Check for Early Stopping:
    - If the validation performance has not improved for a certain number of epochs (patience), stop training.

3. **Output Trained Model:**

- Return the trained detection network model.

VIII. Use Cases and Applications

- The ability to accurately detect artificially colorized images has numerous important applications across various domains:
- Digital Forensics: In legal and investigative contexts, verifying the authenticity of images is crucial. Detecting fake colorization can help identify tampered evidence and ensure the integrity of digital records.
- Journalism: News organizations rely on the credibility of visual information. The technology can assist in verifying the authenticity of images used in news reports, preventing the spread of misinformation.
- Historical Archiving: While artificial colorization can be used to enhance historical images, it's important to distinguish between original and manipulated images for accurate historical documentation.
- Art Authentication: In the art world, determining the authenticity of artwork is essential. The technology can be used to analyze the color characteristics of images of paintings or other artwork to detect potential forgeries or alterations.
- Social Media Monitoring: Social media platforms can use the technology to automatically flag potentially manipulated images, helping to combat the spread of fake news and disinformation.

IX. Future Research Directions

- While the proposed method demonstrates promising results, there are several avenues for future research:
- Improving Robustness: The system's robustness to various types of colorization artifacts and image distortions can be further improved.

- Extending to Other Manipulations: The method can be extended to detect other types of image manipulations beyond colorization, such as splicing or copy-move forgeries.
- Real-time Detection: Developing real-time detection capabilities would enable the system to be used in live applications, such as social media monitoring.
- Explainable AI (XAI): Investigating Explainable AI techniques to provide insights into why the system classifies an image as real or fake can enhance trust and understanding.
- Dataset Expansion: Creating larger and more diverse datasets of real and fake colorized images would help to improve the generalization ability of the model.

X. Conclusion

This paper presents a robust and effective method for detecting artificially colorized images using Channel Difference Maps (CDMs) and a dual-network architecture. The proposed method leverages the unique properties of CDMs to capture color inconsistencies and employs transfer learning to enhance the performance of the detection network. The experimental results demonstrate the effectiveness of the approach, highlighting its potential for various applications in digital image forensics and related fields.
