A Two-Stage Frequency-Based Approach for Detecting AI-Generated Images

Kodithuwakku K.S.
220323N

Botheju W.P.B.
220075E

Arachchi K.A.K.N.
220039A

Abstract

Generative AI models (GANs, diffusion models) are getting very good at making fake images that look real. This makes it hard to tell which images are real and which are AI-made. We build a two-stage system that use frequency-domain analysis and statistical methods to detect AI-generated images. Our approach have: (1) a Frequency-Aware CNN that look at both normal RGB colors and FFT-based mid-frequency features using modified ResNet18, and (2) a statistical checker based on 3D color histogram analysis and KDE to fix low-confidence predictions. We show that frequency-domain features, when we combine with data augmentation (JPEG compression, Gaussian blur), help detect known generator patterns while histogram-based checking helps identify outputs from new generative models we not seen before. Experimental results show that our method work better than spatial-only approaches for generalization. Implementation code and notebooks are available.

1. Introduction

High-quality generative models can now create synthetic images that look almost exactly like real photos. GANs and diffusion models can make realistic faces, scenes, and objects very easily. While these technologies are good for creative work, they also create serious problems for misinformation, identity theft, and make people not trust digital media anymore. We need automated detection systems for content moderation, news checking, and forensic work.

This was developed as a solution for the Kaggle competition "Detect AI vs Human Generated Images". The competition asks participants to build a binary classifier that can tell real photographs apart from AI-generated images, with performance measured using AUC-ROC score on a hidden test set.

Old detection methods that only look at spatial-domain features often don't work well across different generator types. Recent papers show that generative models leave special "fingerprints" in frequency domain [5], which stay there even after normal image changes. But attackers can remove or reduce these fingerprints by filtering frequencies [1], which may be included in the test dataset, so we need strong detection methods.

This work tries to build a practical detection system with these goals:

1. Use frequency-domain artifacts that come from generative models but are harder to remove with simple spatial tricks
2. Mix deep learning features with statistical methods like histogram analysis so we can handle both known and unknown generator types
3. Add data augmentation to make system more robust against fingerprint-removal
4. Make implementation that balance accuracy with speed

2. Problem Statement

We formulate AI-generated image detection as binary classification problem. Given an RGB image $I \in R^{(H \times W \times 3)}$, the goal is predict a label $y \in \{0, 1\}$ where:

- $y = 0$ real (camera-captured or human-created)
- $y = 1$ AI-generated (synthetic) image

2.1 Evaluation Metrics

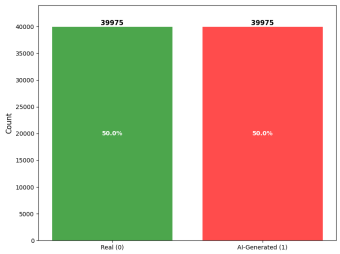
We evaluate detection performance using standard classification metrics:

- F1 Score: Harmonic mean of precision and recall at default threshold, balancing false positives and false negatives. This is the primary metric used by the Kaggle competition.
- AUC-ROC: Area under the receiver operating characteristic curve, measuring how well we can discriminate across all thresholds.
- Accuracy: Overall fraction of correct predictions
- Confusion Matrix: Detailed breakdown of true positives, true negatives, false positives, and false negatives

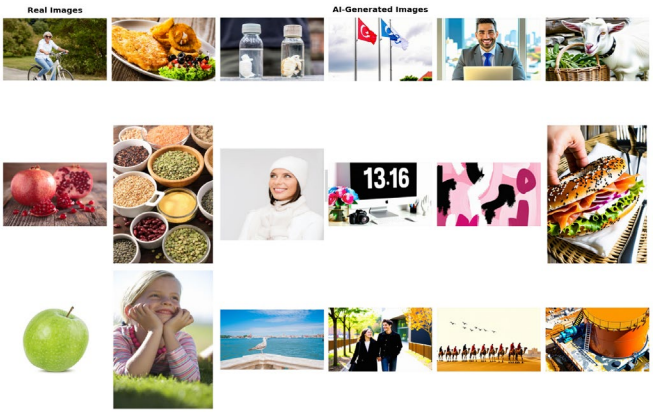
We consider AUC as primary metric since it show performance independent of threshold choice, even though F1 is the metric used in the competition.

2.2 Dataset Characteristics

The Kaggle competition "Detect AI vs Human Generated Images" provide a dataset with 79,950 training images and 5,540 test images. The training set is balanced with exactly 39,975 real images and 39,975 AI-generated images, which means we don't need special techniques for handling class imbalance.



The real images in the dataset come from different sources and show diverse content including people, animals, landscapes, and everyday objects. This variety is important because it mean the detector need to work across different scene types and not just memorize specific patterns from one category.



Dataset Samples: Read Images (Left) vs AI-Generated Images (Right)

The AI-generated images are made by multiple types of generative models, including both GANs and diffusion models. This is a challenging aspect of the dataset because different generators leave different artifacts and fingerprints. Some images might come from older GAN architectures with more obvious problems, while others might be from newer diffusion models that produce very realistic outputs. The detector need to learn features that generalize across these different generator types.

All images in the dataset are RGB color images, but they have different resolutions and aspect ratios. Our preprocessing step resize everything to 256×256 pixels to make a consistent input size for the neural network. The test set labels are hidden, so we can only evaluate final performance by submitting predictions to the competition platform.

3. Literature Review

Our approach is built on several areas of research in AI-generated image detection. First we discuss the deep learning architecture we use as our base model, ResNet-18. Then we look at work on GAN fingerprints that show generators leave unique traces in images. Next we review frequency-domain detection methods that use FFT and DCT to find synthetic artifacts. We also discuss data augmentation strategies that make detectors more robust. Finally we cover histogram-based forensic techniques for image analysis. This review help us understand what works, what doesn't.

3.1 Transfer Learning

He et al. [8] introduced ResNet (Residual Networks), which solve the problem of training very deep neural networks. Before ResNet, deeper networks often performed worse than shallow ones because of vanishing gradients during training. ResNet use skip connections (residual connections) that let gradients flow directly through the network, making it possible to train networks with 50, 101, or even 152 layers. ResNet18, which we use in our work, is a smaller variant with 18 layers that balance accuracy and speed.

We choose ResNet18 as our base model because it already been trained on ImageNet with millions of images, so it learned useful features for recognizing objects and patterns. It's computationally efficient compared to bigger models, which important for practical deployment, and the pretrained weights give us a good starting point, and we can adapt them to our detection task. We modify the first layer to accept our 6-channel input (3 spatial + 3 frequency) and the last layer to output binary predictions.

3.2 GAN Fingerprints

Yu et al. [5] showed that GAN generators leave stable, model-specific fingerprints in generated images. These fingerprints stay across different image patches and can be found in both spatial and frequency domains. Their work showed that CNN trained on fingerprint patterns can tell which GAN made a fake image with high accuracy. This finding make us want to use frequency-domain features, because generator-specific artifacts are often more clear in spectral representations.

However, Wesselkamp et al. [1] found problems in fingerprint-based detection. They showed that attackers can remove or reduce GAN fingerprints by applying frequency filtering, specially by removing high-frequency content. This adversarial view is important for real-world use, because it mean that detectors that only use high-frequency artifacts are weak. Their work directly motivate our augmentation strategy: by training with JPEG compression and Gaussian blur (which remove high-frequency info), we force model to learn more robust mid-frequency representations.

3.3 Frequency and DCT-based Detection

Several recent works used frequency-domain representations for deepfake detection. Tan et al. [4] proposed frequency-domain learning modules that process FFT phase and amplitude separately, showing that frequency-space learning improve cross-generator generalization. Their architecture use FFT→processing→iFFT blocks to extract frequency-aware features, which inspired our FrequencyExtractor design.

Giudice et al. [6] focused on DCT coefficient analysis to find GAN-specific problems. They showed that some DCT coefficients show big differences between real and GAN-generated images, and these differences are somewhat strong to JPEG compression. Their findings help our choice of frequency bands: mid-frequency parts (not too low or too high) have useful information while being less affected by common image changes.

Pontorno et al. [7] extended this work by doing systematic study of which DCT coefficient combinations work best for different generative models (GANs, diffusion models). They showed that carefully picked frequency bands can generalize better than full-spectrum approaches. This work give empirical reason for our band-pass filtering approach with adjustable parameters.

3.4 Data Augmentation

Hendrycks and Dietterich [2] introduced ImageNet-C, a benchmark for testing neural network robustness to common corruptions. They showed that standard ImageNet-trained models are weak when they face JPEG compression, Gaussian blur, noise, and other real-world problems. Their work established that training with corruption augmentations make model more robust without losing clean-data accuracy. We use similar augmentation strategies (JPEG, blur, color jitter) to improve our detector.

3.5 Histogram Based Statistical Analysis

Yang et al. [3] showed that histogram-domain features give extra information to spatial CNN features for image forensic tasks. They showed that color histogram statistics can detect contrast changes and other manipulations that leave small distribution traces. This work motivate our use of 3D color histograms as secondary signal for refinement.

While histogram analysis not new in forensics, its combination with deep learning for generative image detection get less attention. Our approach use histogram-based KDE analysis specifically for refining low-confidence predictions, not as main detection method. This design balance cost with benefit of catching distribution problems that spatial CNNs might miss.

3.6 Gap in Existing Work

Most existing methods focus either on frequency-domain features or statistical analysis, but not both. Also, many approaches don't address robustness to fingerprint-removal attacks explicitly. Our work fill this gap by combining frequency-aware deep learning with statistical refinement, while adding defensive augmentations during training.

4. Proposed Method

Our detection system have two stages. First, a Frequency-Aware CNN makes predictions by looking at both normal image features and frequency patterns. Then, a histogram checker fixes uncertain predictions. This helps catch both familiar generators and new AI models.

4.1 Frequency-Aware CNN Architecture

The FrequencyAwareCNN process images by looking at regular RGB colors and frequency information together. It have three main parts.

The frequency extractor use FFT (Fast Fourier Transform) to convert images into frequency space, then apply a band-pass filter keeping only mid-frequency components (radius 30-100). We chose this range because research [7] showed GAN artifacts show up most clearly here. High frequencies get removed by JPEG compression, and low frequencies don't help much. After filtering, we convert back to normal space and normalize values to [0,1], giving us three frequency channels.

Next we combine these three frequency channels with the three original RGB channels to make a 6-channel input. This let the network learn from both types of information at once.

For the base model, we use ResNet18 [8] pretrained on ImageNet. We modify the first layer to accept 6 channels instead of 3, initializing weights by copying the pretrained weights twice (once for spatial, once for frequency). We also replace the final layer to output a single value for binary classification. We train with Binary Cross-Entropy loss and Adam optimizer using learning rate 0.00003 to avoid forgetting pretrained features.

4.2 Data Augmentation

We apply random changes to training images to make the model robust. Research [1] showed attackers can remove GAN fingerprints by filtering, so we need to handle this.

We use four augmentations: JPEG compression (50% probability, quality 30-100), Gaussian blur (50% probability with random strength), color jitter (70% probability for brightness/contrast/saturation), and horizontal flip (50% probability). Training use all augmentations together, but testing use none.

4.3 Histogram-Based Refinement

For uncertain predictions (probability near 0.5), we use a statistical approach with color histograms.

We compute a 3D color histogram counting pixel combinations across RGB values using 256 bins per channel. The mean of these values give a single number representing the image's color distribution. Real and AI images have different patterns here.

During training, we compute histograms for about 5,000 real images and use Kernel Density Estimation (KDE) to learn the typical distribution. We find the peak value and set a 99% confidence interval to define "normal" for real images.

During testing, we only check histograms for uncertain images (probability < 0.99). If the histogram value fall inside the normal range, we classify as real, otherwise as AI-generated. This save computation on confident predictions.

4.4 Implementation Details

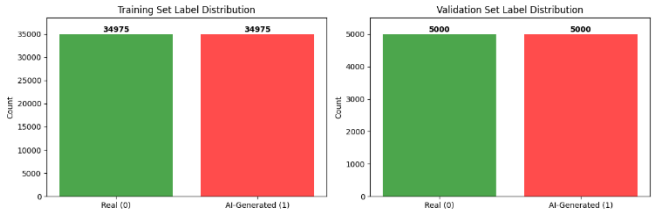
We implemented everything in PyTorch version 2, using torchvision to load the pretrained ResNet18 model. Images get loaded with PIL, resized to 256×256 pixels, and converted to tensors with values

between 0 and 1. We process images in batches of 16 at a time, with error handling so corrupted files don't crash the whole program.

For the histogram computation part, we use Python's multiprocessing with 4 workers to compute multiple histograms at same time (the Kaggle environment CPU has 4 cores) which make it faster. We set random seeds (1 for both numpy and PyTorch) at the start so our results are reproducible.

4.5 Dataset and Preprocessing

We split training data into 69,950 for training and 10,000 for validation. Images get loaded with PIL, converted to RGB, resized to 256×256 with resampling, and normalized to [0, 1]. Training images are then augmented.

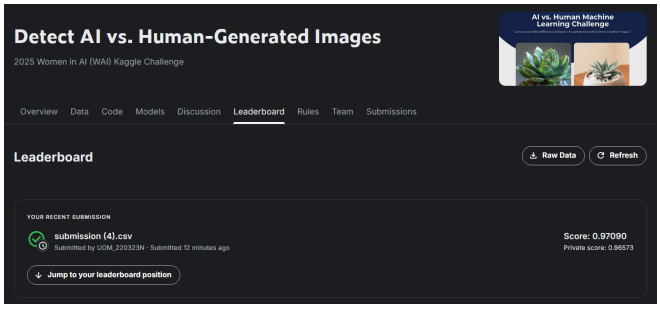


Training and Validation set label distribution

5. Results

5.1 Competition Results

Our final submission achieved a public leaderboard score of 0.97090 and private leaderboard score of 0.96573.



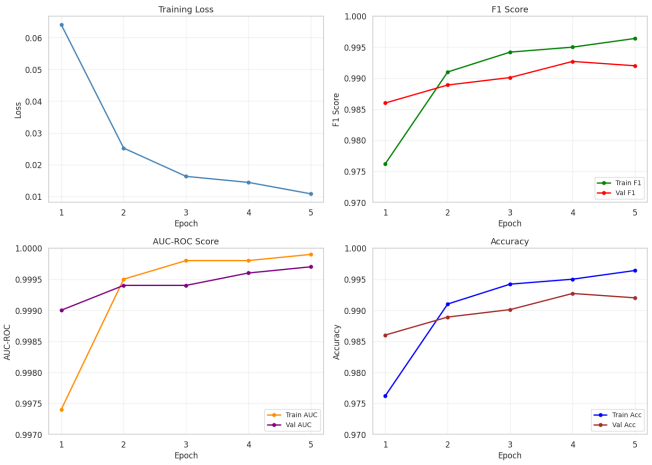
5.2 Model Performance

5.2.1 Final Model Results

Split	Loss	F1 Score	AUC	Accuracy
Train	0.0108	0.9964	0.9999	0.9964
Validation	-	0.9920	0.9997	0.9920

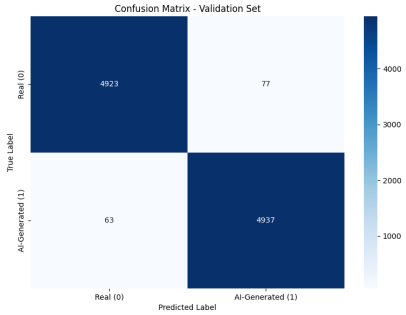
5.2.2 Training Metrics

The model is trained for 5 Epochs.



Training and validation performance metrics over epochs show improvement and convergence.

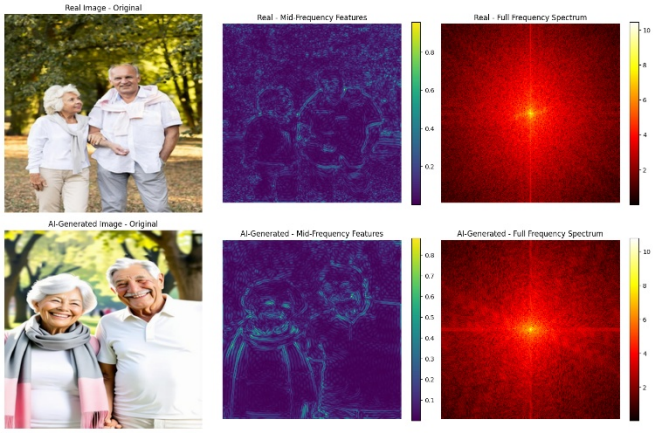
5.2.3 Confusion Matrix



Heatmap showing true positives, false positives, true negatives, and false negatives for full system on validation set.

5.3 Visual Analysis

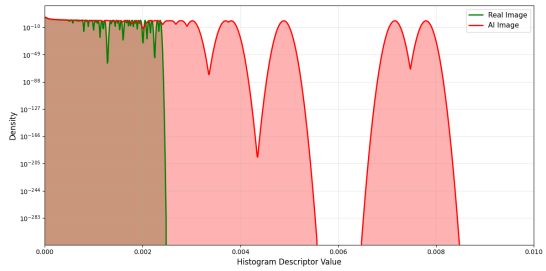
5.3.1 Frequency Visualization



Side-by-side comparison of real vs AI-generated images showing: (left) original, (center) mid-frequency features, (right) full spectrum.

Real images have smooth patterns while GAN images show grid-like artifacts.

5.3.2 Histogram Distribution



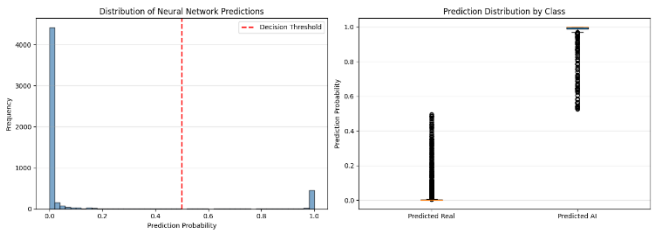
KDE curves for 3D color histogram descriptors. Real and AI have different distributions, explaining why statistical refinement helps.

Real images have smooth frequency patterns. GAN images have grid artifacts in mid-frequencies. Diffusion models show different patterns than GANs.

System struggles with compressed real photos (compression add fake-looking artifacts), high-quality diffusion outputs (too realistic), and hybrid images (mix of real and AI edits).

Histogram distributions are different for real vs AI images. Real images cluster around certain color patterns. AI images have different peaks.

5.3.3 CNN Prediction Distribution



Histogram and box plot showing distribution of neural network prediction probabilities on test set. Left plot show histogram with decision threshold at 0.5. Right plot show box plots separating predicted real vs AI images.

6. References

[1] V. Wesselkamp, K. Rieck, D. Arp, and E. Quiring, "Misleading Deep-Fake Detection with GAN Fingerprints," 2022. https://arxiv.org/abs/2205.12543

[2] D. Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," 2019. https://arxiv.org/abs/1903.12261

[3] P. Yang, R. Ni, Y. Zhao, G. Cao, and W. Zhao, "Robust Contrast Enhancement Forensics Using Pixel and Histogram Domain CNNs," IEEE Trans. Circuits Syst. Video Technol., 2019. https://arxiv.org/abs/1803.04749

[4] C. Tan, Y. Zhao, S. Wei, G. Gu, P. Liu, and Y. Wei, "Frequency-Aware Deepfake Detection: Improving Generalizability through Frequency Space Learning," 2024. https://arxiv.org/abs/2403.07240

[5] N. Yu, L. Davis, and M. Fritz, "Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints," 2019. https://arxiv.org/abs/1811.08180

[6] O. Giudice, L. Guarnera, and S. Battiato, "Fighting deepfakes by detecting GAN DCT anomalies," J. Imaging, vol. 7, no. 8, p. 128, 2021. https://arxiv.org/abs/2101.09781

[7] O. Pontorno, L. Guarnera, and S. Battiato, "On the Exploitation of DCT-Traces in the Generative-AI Domain," 2024. https://arxiv.org/abs/2402.02209

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016. https://arxiv.org/abs/1512.03385