Detecting AI-Generated Art Using Machine Learning

Michal Shasha 325763498

Daniel Tsadik 209307727

Machine Learning

Ariel University

July 2025

1. Introduction

In recent years, the proliferation of AI-generated content has raised new challenges in the fields of art, media, and digital verification. Among these challenges lies the growing difficulty in distinguishing between artwork created by human artists and those generated by artificial intelligence (AI) systems. As generative models such as DALL·E, Midjourney, and Stable Diffusion continue to advance, the visual gap between "real" and AI-generated art is narrowing, making manual classification increasingly unreliable.

This project addresses the classification problem of identifying whether a given image is a piece of human-created art or one produced by an AI model. This task is not only of

academic interest but also holds practical implications in domains such as copyright enforcement, content moderation, digital forensics, and cultural preservation.

Our work involves designing a machine learning pipeline that can learn to distinguish between the two classes (RealArt and AiArtData) based solely on features extracted from the image data. The goal is to build and evaluate classification models that operate on visual characteristics of the artworks and attempt to detect subtle stylistic patterns that differentiate human-made from AI-made content.

We begin by constructing a labeled dataset of images collected from both real-world art repositories and AI art generators. These images are processed to extract meaningful feature representations, which serve as inputs to the machine learning algorithms.

This project represents the first stage in a broader investigation into the feasibility of automated AI art detection. In the following sections, we describe the dataset, preprocessing steps, feature extraction methods, model selection, evaluation metrics, and experimental findings.

2. Dataset Description

The dataset, titled "AI Generated Images vs Real Images", consists of a total of 975 labeled images, divided into two categories:

AiArtData: 539 images generated by artificial intelligence models.

RealArt: 436 images of artwork created by human artists.

Each image is stored in a corresponding folder based on its class label. A metadata CSV file maps each image filename to its label, enabling efficient loading and processing. This dataset forms the basis for training and evaluating binary classifiers that aim to distinguish between AI-generated and real human-created art.

3. Feature Extraction

To enable machine learning algorithms to differentiate between real and AI-generated artworks, we must first represent each image as a fixed-length numerical feature vector. This transformation is performed in our project by the script extract_features.py, located in the Features/ directory.

The script processes all images in the dataset originally stored in two subdirectories: Data/RealArt/ and Data/AiArtData/. It reads each image, resizes it to a uniform resolution of 128×128 pixels, and then converts it into a histogram-based representation using color information.

Specifically, we compute a 3D color histogram in the HSV color space, dividing the hue, saturation, and value dimensions into 8 bins each. The resulting feature vector for each image consists of 512 values (8×8×8), which reflect the distribution of pixel colors across the image. These features are normalized to ensure comparability across images.

The output of the feature extraction process is saved to a file named features.csv, which contains one row per image and includes:

- A filename identifier
- The original label (RealArt or AiArtData)
- The 512-dimensional normalized feature vector

To ensure a balanced dataset for training, we removed excess samples from the larger class (AI-generated images). Specifically, we downsampled the AI class to match the number of available RealArt images, resulting in a final dataset of **436 images per class**, for a total of **872 labeled samples**.

The feature extraction as seen in the terminal:

```
⚖️ Balancing classes to 434 images each
Processing RealArt: 100%|██████████| 434/434 [00:09<00:00, 44.18it/s]
Processing AiArtData: 100%|██████████| 434/434 [00:06<00:00, 64.61it/s]
✅ Features saved to Features\features.csv
```

This feature extraction step ensures that all subsequent models operate on a uniform and informative representation of the visual data, while maintaining class balance for fair evaluation.

4. Evaluation of Supervised Models

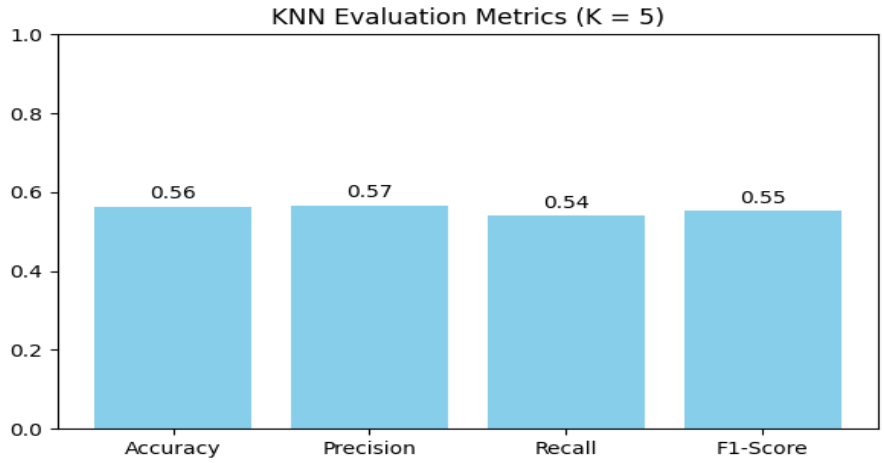
4.1 K-Nearest Neighbors (KNN) – Baseline and Cross-Validated Versions

As a starting point, we implemented the K-Nearest Neighbors (KNN) classifier, a simple, intuitive, and widely used model that serves as a strong baseline in many classification tasks. KNN makes predictions based on the majority label among the *k* closest samples in the training set, using a distance metric (typically Euclidean distance) in the feature space. Despite its simplicity, KNN can be surprisingly effective, particularly when combined with proper feature extraction and parameter tuning.

Initial Version: Hold-Out Validation

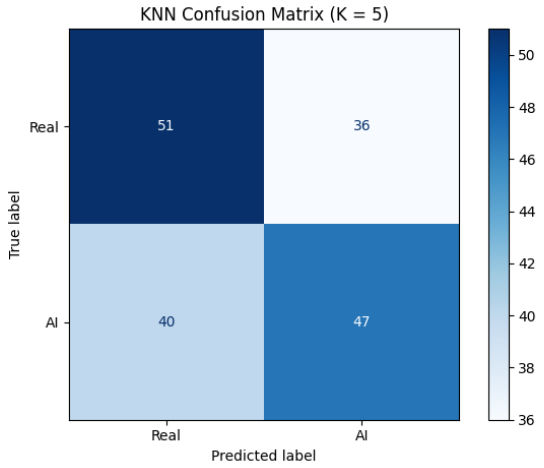
In the initial implementation, the dataset was randomly split into training and test subsets (80% / 20%) using stratified sampling to preserve label proportions. We evaluated a range of *k* values (from 1 to 20), training a separate model for each value and computing its performance on the test set. The goal was to identify the value of *k* that yielded the highest F1 score — a balanced metric that combines precision and recall. The best value found was used to train a final model and generate evaluation outputs.

Based on the evaluation results of the hold-out validation with K = 5, the KNN classifier achieved moderate performance across all major metrics. The overall accuracy on the test set was **56%**, with relatively balanced precision and recall across the two classes. Specifically, the model demonstrated a precision of **0.56** and recall of **0.59** for *RealArt*, and a precision of **0.57** and recall of **0.54** for *AiArtData*. These results indicate that the model performs similarly across both classes, with a slight tendency to better identify real artwork. The F1-scores, which provide a harmonic mean of precision and recall, were **0.57** for *RealArt* and **0.55** for *AiArtData*.



The confusion matrix reveals the model's classification tendencies: while it correctly identified **51** real artworks and **47** AI-generated artworks, it also misclassified **36** real

images as AI and **40** AI images as real. These relatively high misclassification rates reflect the inherent difficulty of the task, especially given the visual similarity between real and AI-generated images.



Despite its simplicity, the KNN model provides a valuable first benchmark. The moderate results suggest that while some distinctive patterns may exist in the feature space, they are not trivially separable using instance-based learning alone. This motivates the need to explore more sophisticated classifiers in subsequent experiments.

All plots and output files (e.g., classification report, confusion matrix, bar charts) were saved in the directory:

Results/KNN/

Improved Version: 5-Fold Cross-Validation

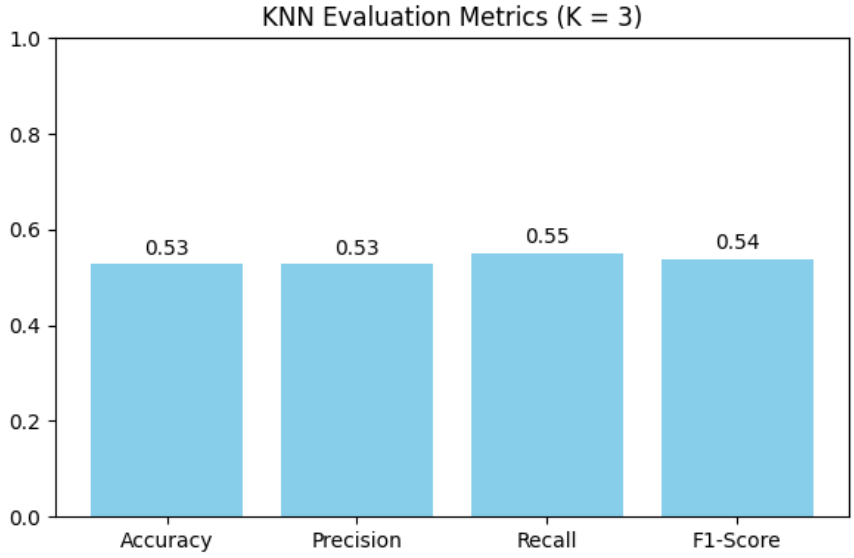
To improve robustness and reduce variance in model selection, we developed an enhanced version of the KNN classifier using 5-fold cross-validation on the entire dataset. For each value of K (from 1 to 20), we trained and evaluated the classifier across five stratified folds, then computed the mean accuracy and F1 score. The value of K that achieved the highest average F1 score was selected as the optimal parameter, and a final evaluation was performed on a separate hold-out test set.

While this version offers a more reliable approach than a single train-test split, it introduces a new source of randomness due to the shuffling of data inside the cross-validation procedure. Because of this, multiple runs of the same script may result in different values of K being chosen, and subsequently, different test results. This behavior reflects the inherent sensitivity of KNN to small data variations, especially in relatively small datasets such as ours.

In the current run, the selected value was K=3, and the final model yielded an accuracy of **53%**, precision of **0.53**, recall of **0.55**, and F1-score of **0.54**. These results were slightly worse than the ones obtained in the hold-out version, suggesting that while cross-validation reduces overfitting, it does not necessarily improve performance in our case.

Best K = 3

	precision	recall	f1-score	support
RealArt	0.53	0.51	0.52	87
AiArtData	0.53	0.55	0.54	87
accuracy			0.53	174
macro avg	0.53	0.53	0.53	174
weighted avg	0.53	0.53	0.53	174



All results for the cross-validated KNN version were stored in:

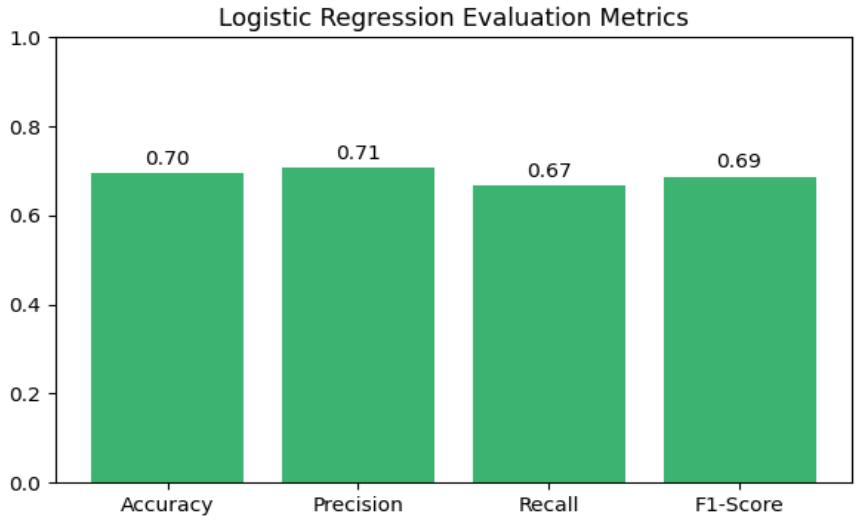
Results/KNN_CV/

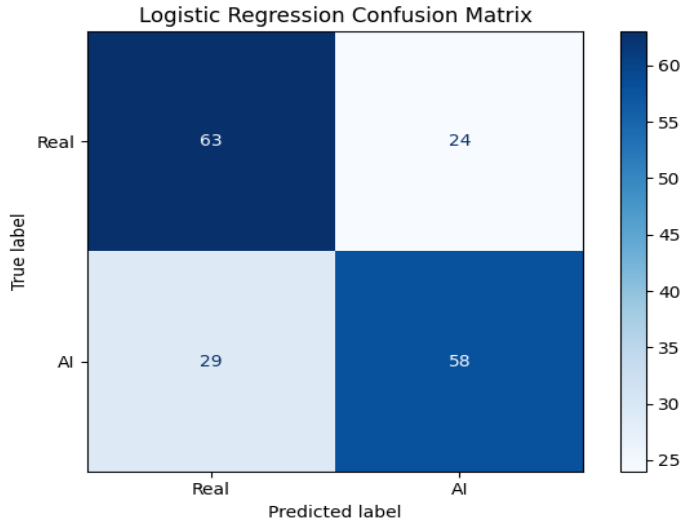
Both versions of the KNN classifier provided useful initial insights into the dataset. The hold-out version achieved stronger overall performance, likely due to benefiting from a favorable random train-test split. In contrast, the cross-validated version offered a more statistically reliable approach by averaging over multiple folds, but resulted in lower accuracy and F1 scores. This highlights a common trade-off in model evaluation: cross-validation improves robustness and generalizability, but may underperform when data is limited or when models are sensitive to small variations. Overall, these two versions laid the groundwork for deeper analysis and more advanced models later in the project.

4.2 Logistic Regression

Following the KNN-based experiments, we trained a Logistic Regression classifier using the same extracted features and hold-out split. Unlike KNN, which relies on local distance-based voting, Logistic Regression attempts to find a global linear decision boundary that best separates the two classes. This makes it a useful next step in assessing whether a linear model can better exploit the patterns embedded in our feature space.

The model achieved **an accuracy of 70%** and an **F1-score of 0.69**, significantly outperforming both KNN variants. The confusion matrix showed relatively balanced performance: it correctly classified **63 out of 87** RealArt images and **58 out of 87** AiArtData images. These results suggest that the feature representations, while not linearly separable in a perfect sense, do contain enough structure for a linear classifier to capture meaningful distinctions between the two classes.



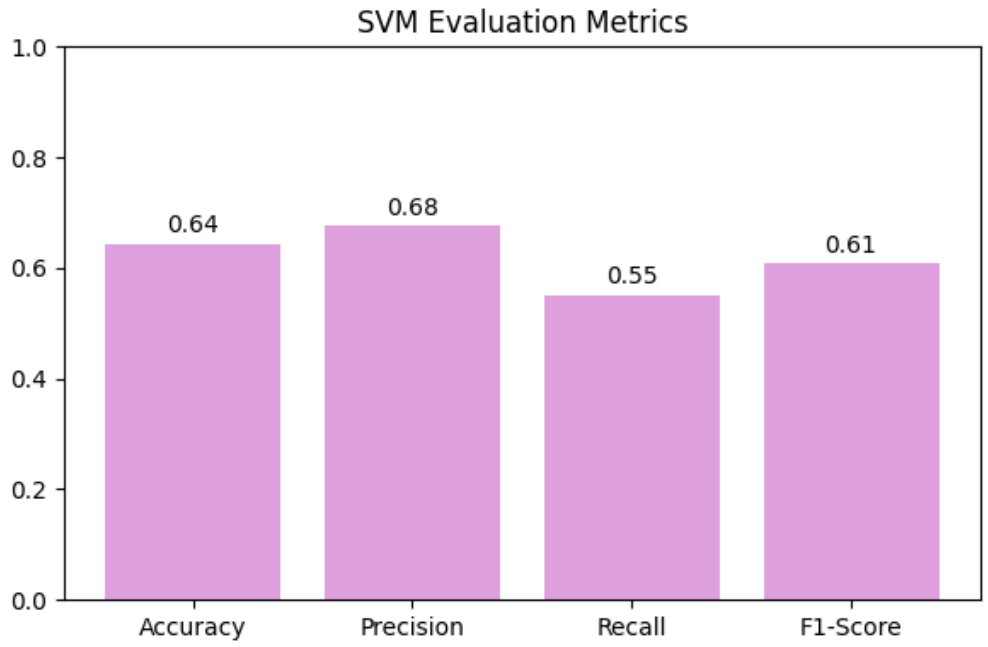


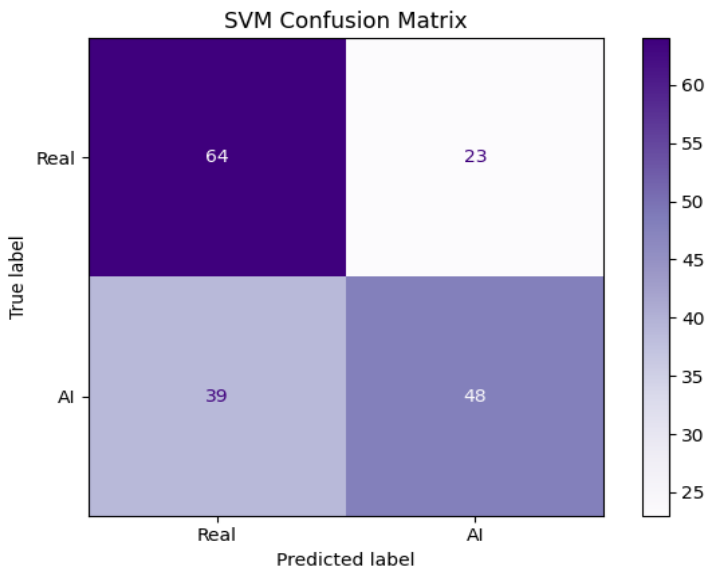
We hypothesize that the improvement in performance stems from the model's ability to learn **global discriminative patterns** across the entire dataset, rather than relying on local neighbor distributions which can be sensitive to noise or overlapping regions. Additionally, Logistic Regression tends to generalize better in high-dimensional spaces when the amount of training data is moderate, which aligns with our scenario.

This result reinforces the idea that even relatively simple linear models can be effective in visual classification tasks when paired with well-designed feature extraction techniques. It also sets a strong baseline for the next stages of the project, where we will explore more expressive models capable of capturing non-linear relationships.

4.3 SVM

The Support Vector Machine (SVM) model yielded moderate results, with an overall accuracy of 0.64. While this marks an improvement over both versions of the KNN model, it falls short of the performance achieved by logistic regression (accuracy of 0.70). The precision for AI-generated art was relatively high (0.68), indicating that the model was cautious when labeling an image as AI. However, recall was significantly higher for RealArt (0.74) than for AiArtData (0.55), meaning that the model was better at identifying human-made artworks but tended to miss many AI-generated ones.



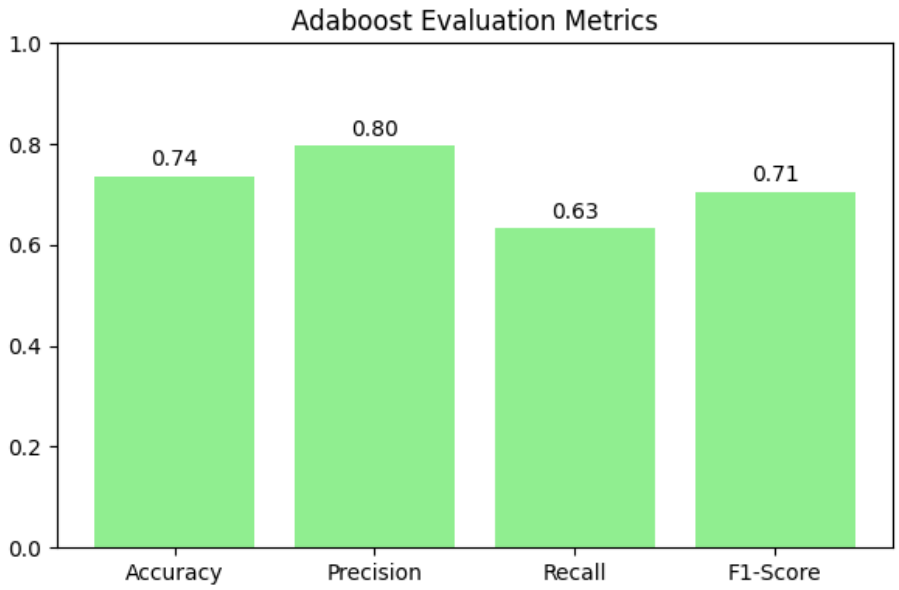


The F1-scores were fairly balanced across the two classes (0.67 for Real, 0.61 for AI), suggesting a somewhat consistent performance despite the uneven recall. Compared to logistic regression, SVM showed slightly lower scores across all metrics. This might indicate that the SVM's linear decision boundary does not fully capture the complexity of the data. Future enhancements could include exploring non-linear kernels to better model intricate patterns in the feature space.

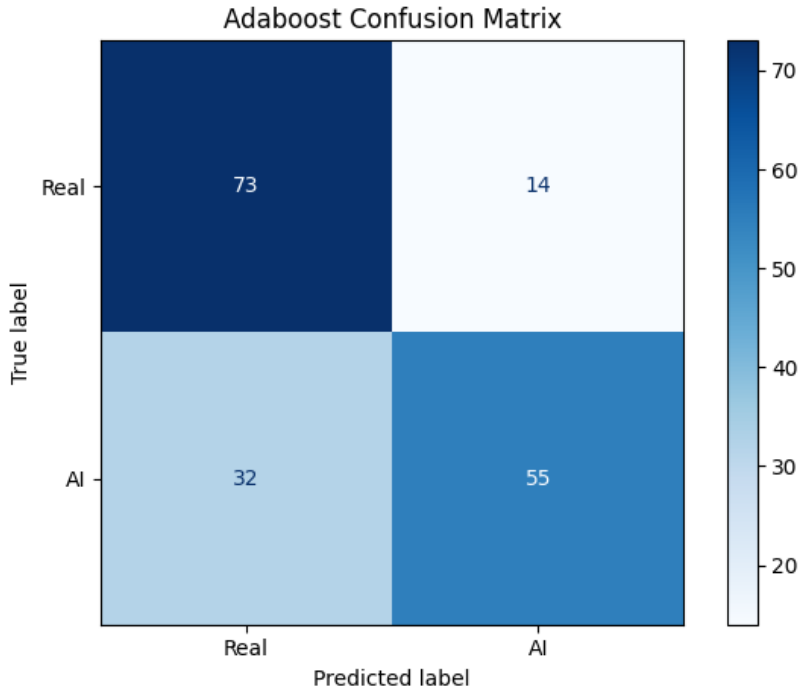
In summary, while SVM demonstrated decent classification ability, it was less robust than logistic regression and exhibited a clear imbalance in detecting the two classes.

4.4 AdaBoost

The AdaBoost model yielded the strongest performance among all traditional (non-neural) classifiers tested in this project. By iteratively combining multiple weak learners—specifically, shallow decision trees—into a weighted ensemble, AdaBoost effectively captured subtle distinctions between real and AI-generated artworks. The resulting classifier achieved an overall accuracy of 74%, with a notably high precision of 0.80 and F1-score of 0.71. This suggests that the model is particularly adept at minimizing false positives, especially when predicting AI-generated images.



The confusion matrix further highlights this strength: the model correctly classified 73 out of 87 real images, and 55 out of 87 AI images. While it struggled slightly more with recall on the AI class (0.63), it maintained a strong balance across precision and recall. Compared to earlier models like logistic regression or KNN, AdaBoost demonstrated both improved discrimination power and greater overall stability, without relying on heavy hyperparameter tuning.



As with all previous classifiers, all the results, including the classification report, confusion matrix, and performance bar charts—were stored in the directory: Results/AdaBoost/

5. Dimensionality Reduction and Data Analysis with PCA

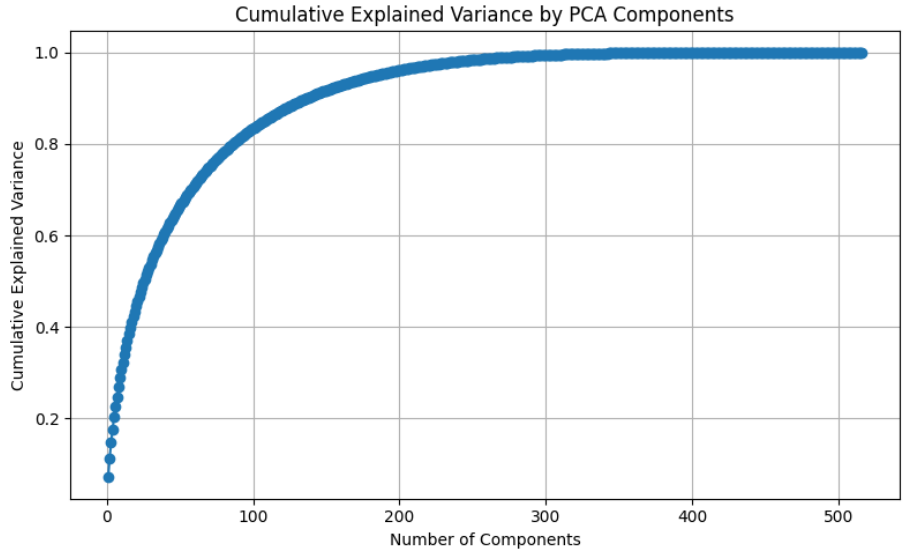
To better understand the structure of the extracted features and investigate the separability of real and AI-generated images in the feature space, we applied **Principal Component Analysis (PCA)** - a widely used linear dimensionality reduction technique. Unlike the supervised models discussed previously, PCA is an **unsupervised method** that aims to project high-dimensional data into a lower-dimensional space while preserving as much variance (i.e., information) as possible.

PCA works by identifying **principal components**, which are linear combinations of the original features that capture the directions of maximum variance in the data. The first principal component (PC1) captures the most variance, the second (PC2) captures the most remaining variance orthogonal to PC1, and so on. These components can reveal latent structure in the data and are particularly useful for visualization and interpretability.

Explained Variance Analysis

The first step was to analyze how much variance is explained by each component. We plotted the explained variance ratio of the top 20 components, as shown in the following bar chart:

Figure: Explained Variance by Principal Component

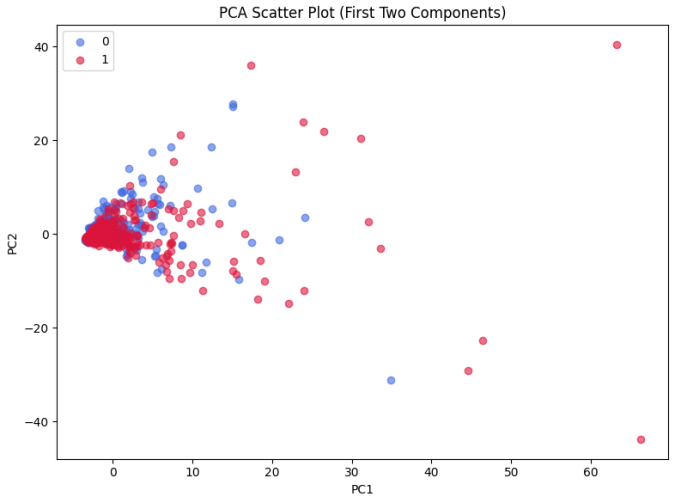


The chart shows that the first few components (especially PC1) explain a large portion of the total variance, while later components contribute significantly less. This suggests that much of the important structure in the dataset can be captured using a relatively small number of components, which justifies dimensionality reduction.

Visualizing the Data in 2D PCA Space

We then projected the data onto the two most informative dimensions: PC1 and PC2. This allowed us to visualize each image (regardless of its original dimensionality) as a point in a 2D space. The resulting scatter plot is presented below, with each point colored according to its label (real or AI-generated):

📌 **Figure: PCA Projection (PC1 vs PC2)**

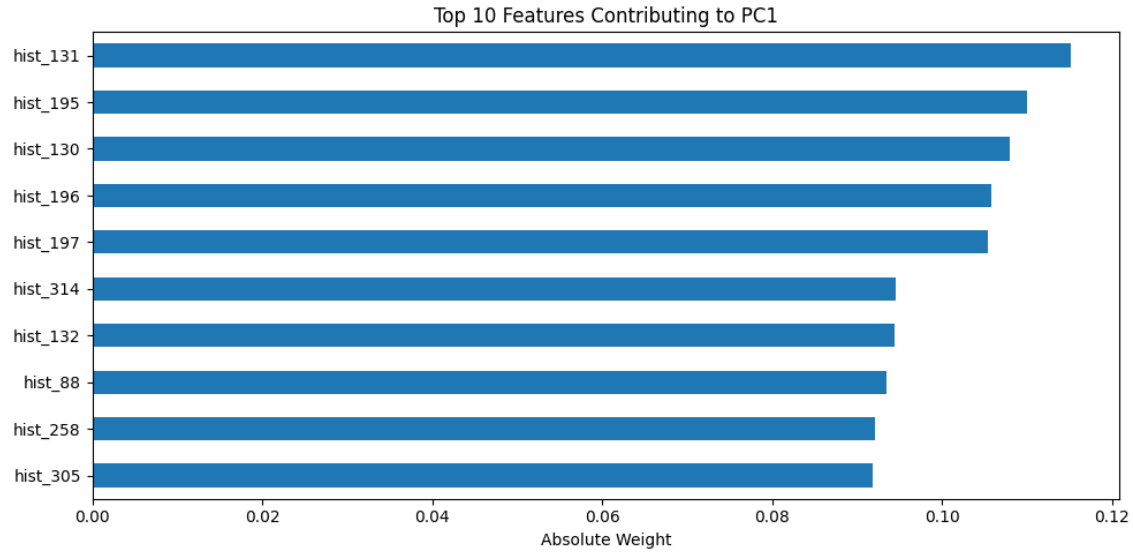


The plot reveals a **partially separable structure**: while the two classes are not linearly separable, there is a degree of clustering. This indicates that some latent structure exists in the data, which PCA was able to uncover. Notably, many real images tend to cluster in different regions of the space compared to AI-generated ones — though with some overlap.

Identifying the Most Influential Features

To interpret the meaning of the principal components, we examined the **top contributing features to PC1**. These are the original features (e.g., color histogram bins) with the largest absolute coefficients in the linear combination that defines PC1. The top 10 such features are shown below:

📈 Figure: Top 10 Feature Contributions to PC1



We also saved the raw data for these contributions in a separate CSV file (top_pc1_features.csv), which can support further interpretability or feature selection efforts.

Interpretation and Discussion

Overall, the PCA analysis confirms that:

- The feature space contains meaningful variance aligned with the labels.
- A reduced-dimensional embedding (e.g., 2D via PC1 and PC2) retains enough structure to **partially distinguish** between real and AI-generated artworks.
- Certain original features strongly contribute to the direction that best separates the data, which may guide future feature engineering or model design.

This analysis provides a **visual and statistical understanding** of the data distribution and suggests that classification may be feasible, albeit non-trivial. The insight that the data is not trivially separable supports the performance trends observed in earlier supervised models (e.g., moderate F1 scores in KNN and logistic regression).

All results were automatically saved to the directory:

PCA/

├── explained_variance.png

├── pca_scatter_plot.png

├── pc1_top_features.png

└── top_pc1_features.csv

6. Addressing Research Questions

We analyze which models perform best at detecting AI-generated art and discuss whether visual features alone are sufficient for this classification.

7. Optional Interface

If an interactive tool was developed, a brief explanation of its usage and screenshots or UI description can be included here.

8. Conclusion and Future Work

We summarize our findings, list limitations, and suggest possible improvements such as CNN-based methods or using more abstract features.

References

Use APA / IEEE / Chicago style depending on requirement.