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.

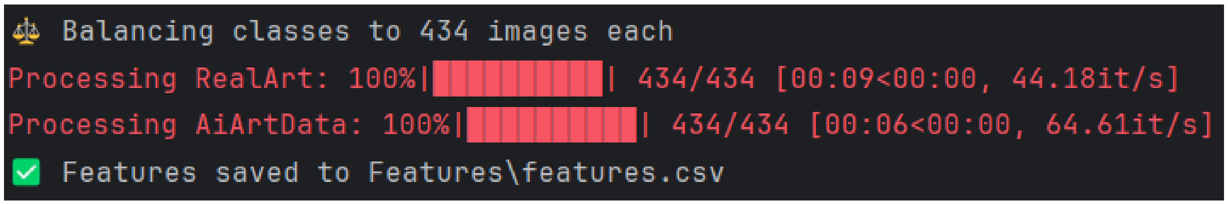
Specifically, we compute a 3D color histogram in the HSV color space, dividing the hue, saturation, and value dimensions into 8 bins each. The resulting feature vector for each image consists of 512 values (8×8×8), which reflect the distribution of pixel colors across the image. These features are normalized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

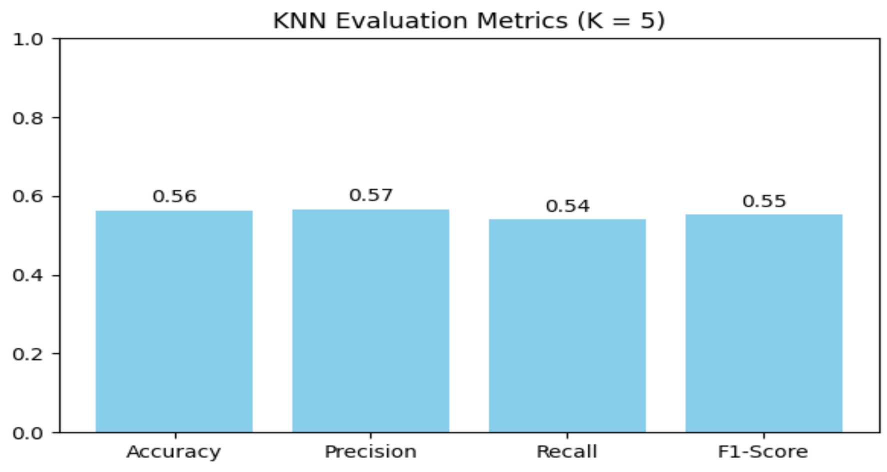
4.1 K-Nearest Neighbors (KNN) – Baseline and Cross-Validated Versions

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.

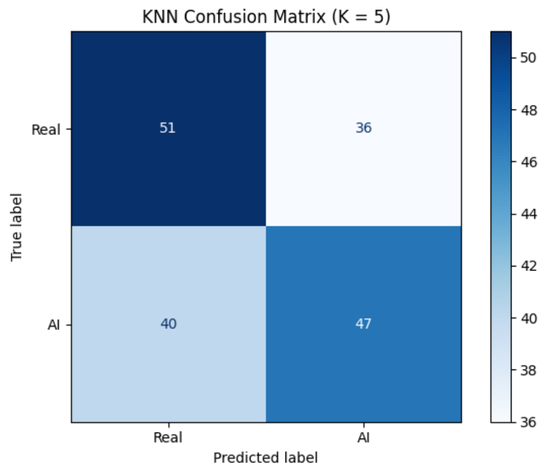
Initial Version: Hold-Out Validation

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/

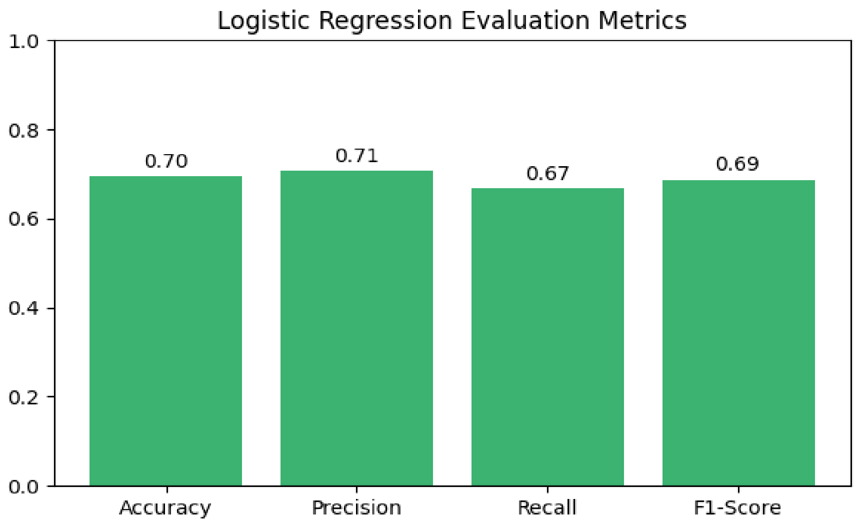
5-Fold Cross-Validation Version
We also experimented with 5-fold cross-validation to select the optimal value of K in KNN. However, this approach did not improve results compared to the simpler hold-out split, with accuracy and F1-score remaining lower. This may be due to the small dataset size and KNN's sensitivity to variations between folds, which can lead to inconsistent performance when data is limited.

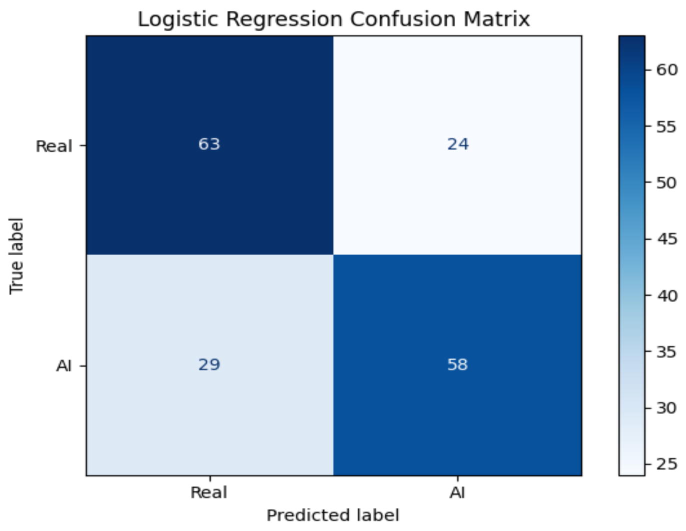
PCA-enhanced Version
We applied PCA before KNN to reduce the feature space to approximately 30 components (explaining ~95% of the variance). This significantly improved results: accuracy increased from ~52% to ~66%, and F1-score improved similarly. This suggests that PCA successfully removed noise and revealed a clearer structure for distance-based classification.

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.





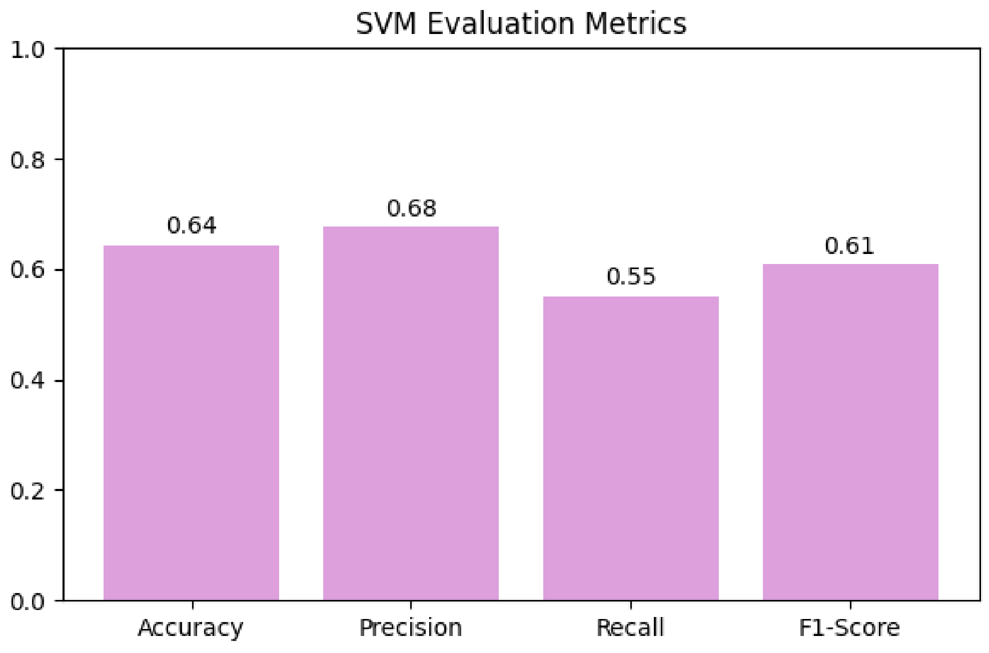
We hypothesize that the improvement in performance stems from the model's ability to learn **global discriminative patterns** across the entire dataset, rather than relying on local neighbor distributions which can be sensitive to noise or overlapping regions. Additionally, Logistic Regression tends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

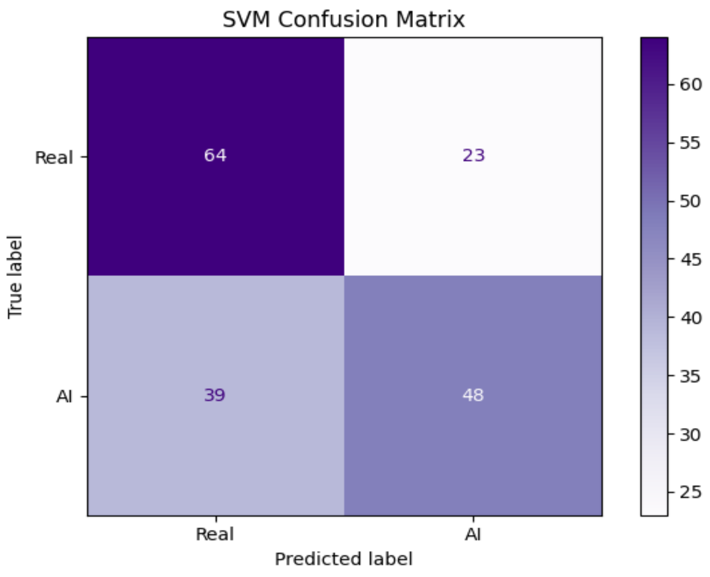
This result reinforces the idea that even relatively simple linear models can be effective in visual classification tasks when paired with well-designed feature extraction techniques. It also sets a strong baseline for the next stages of the project, where we will explore more expressive models capable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

PCA Experiment:
We also tested Logistic Regression with PCA preprocessing. This dimensionality reduction did not improve results. The accuracy and F1-score decreased slightly to around 66%, suggesting that PCA may have removed useful discriminative information for this linear model.

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.





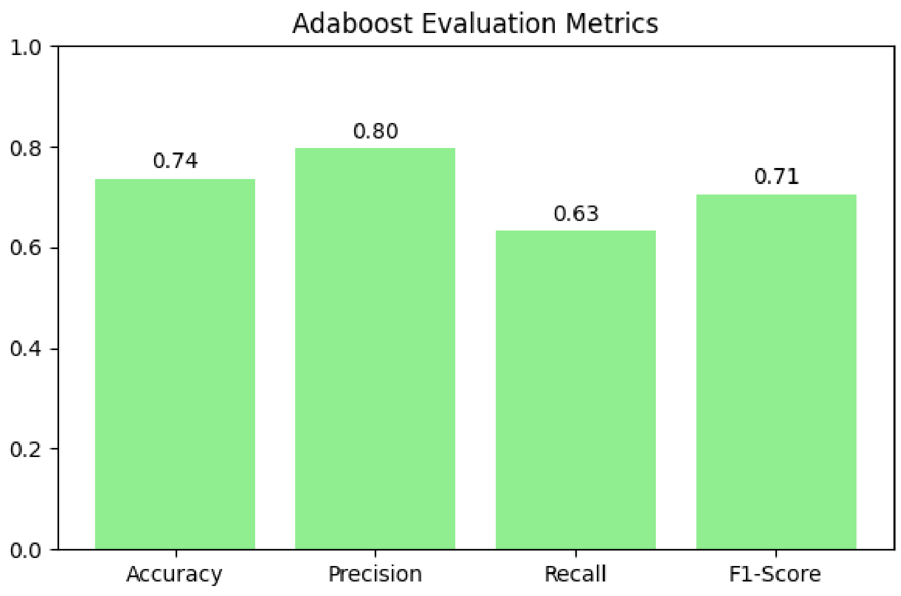
The F1-scores were fairly balanced across the two classes (0.67 for Real, 0.61 for AI), suggesting a somewhat consistent performance despite the uneven recall. Compared to logistic regression, SVM showed slightly lower scores across all metrics. This might indicate that the SVM's linear decision boundary 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.

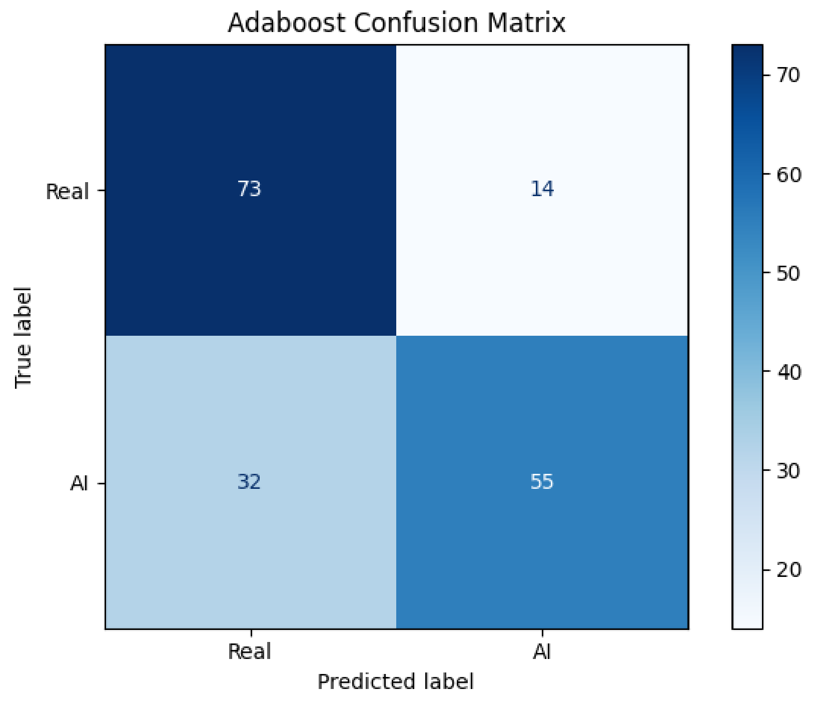
PCA Experiment
We evaluated SVM with PCA preprocessing (about 30 components). Performance was similar to the original version (accuracy ~63–65%, F1-score ~63–65%), suggesting that PCA did not improve separability for the linear decision boundary of SVM.

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, all were stored in the directory: Results/AdaBoost/

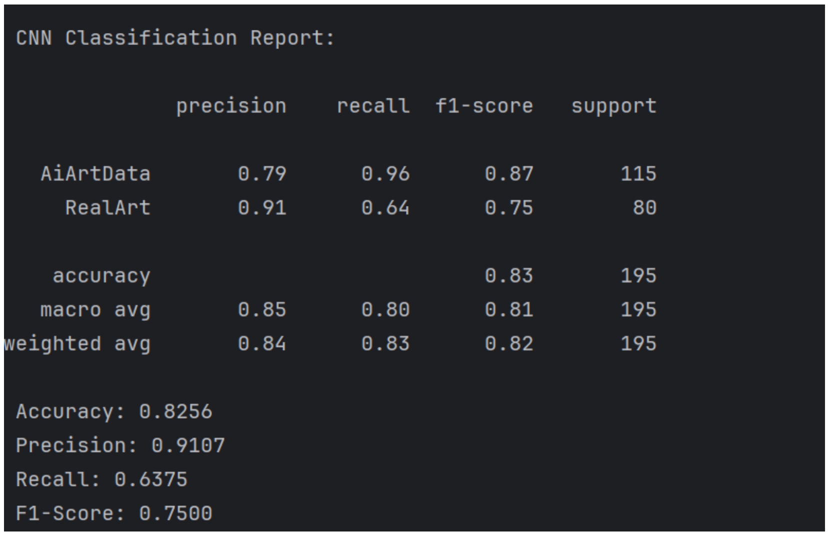
PCA Experiment
We also tried adding PCA before training AdaBoost. However, performance dropped noticeably: accuracy fell from ~74% to ~64%. This indicates that AdaBoost benefits from the full, high-dimensional feature space, where weak learners can exploit subtle variations that PCA may remove.

4.5 Convolutional Neural Network (CNN) Classifier

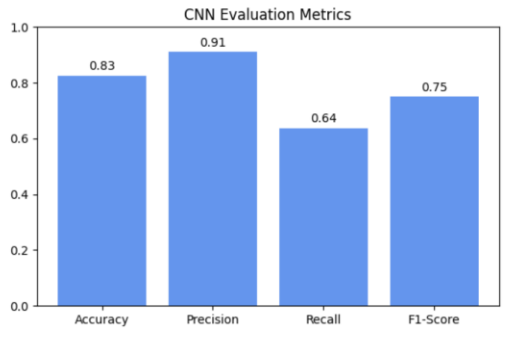
A Convolutional Neural Network (CNN) was implemented to classify the raw images directly, instead of relying on precomputed color histogram features. A pretrained ResNet18 architecture was used, with its final layer modified to output two classes: RealArt and AiArtData. All layers were fine-tuned on the dataset to better adapt the model's learned features to this specific task.

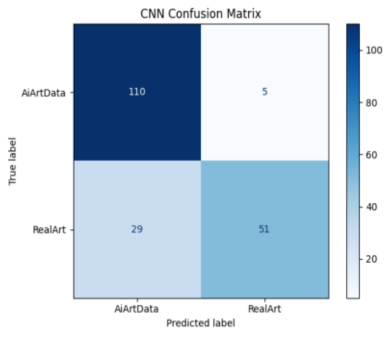
To improve robustness and reduce overfitting, strong data augmentation techniques were applied, including random horizontal flips, rotations, color jitter, affine transformations, and perspective distortion. A learning rate scheduler was employed to reduce the learning rate during training, allowing for better convergence. Early stopping with a patience of 10 epochs was used based on validation loss, with the best-performing model weights saved automatically.

The CNN was evaluated on the held-out test set (20% of the data). It achieved approximately **82.5% test accuracy**, outperforming all previous models (including AdaBoost at 74%). This result demonstrates that learning directly from pixel data enables the model to capture the subtle visual differences between AI-generated and human-created art.



```
CNN Classification Report:

             precision    recall  f1-score   support

   AiArtData       0.79      0.96      0.87       115
     RealArt       0.91      0.64      0.75        80

    accuracy                           0.83       195
   macro avg       0.85      0.80      0.81       195
weighted avg       0.84      0.83      0.82       195

Accuracy: 0.8256
Precision: 0.9107
Recall: 0.6375
F1-Score: 0.7500
```





5. Summary of the Models

Among all the models we tested, the Convolutional Neural Network (CNN) performed the best with around 82.5% accuracy. It worked well because it learned detailed visual patterns directly from the raw images. AdaBoost was the strongest of the traditional models, reaching about 74% accuracy by combining many simple decision trees to pick up subtle differences in color histograms. Simpler models like KNN and Logistic Regression did fairly well (around 56–70% accuracy) but were limited because they relied only on handcrafted features and simpler decision boundaries. Overall, there was a clear difference between real and AI-generated art, but some AI images were hard to classify because they closely mimicked human styles. Errors often happened with minimalist or abstract art that didn't have clear color patterns, showing that AI art is getting realistic enough to be confusing. Using PCA also had mixed results: it helped KNN by reducing noise and making distances between samples clearer in fewer dimensions, but didn't help much, or even hurt: Logistic Regression, SVM, and AdaBoost. That's because PCA can remove low-variance features that actually help separate the classes, especially for complex models like AdaBoost that benefit from high-dimensional details.

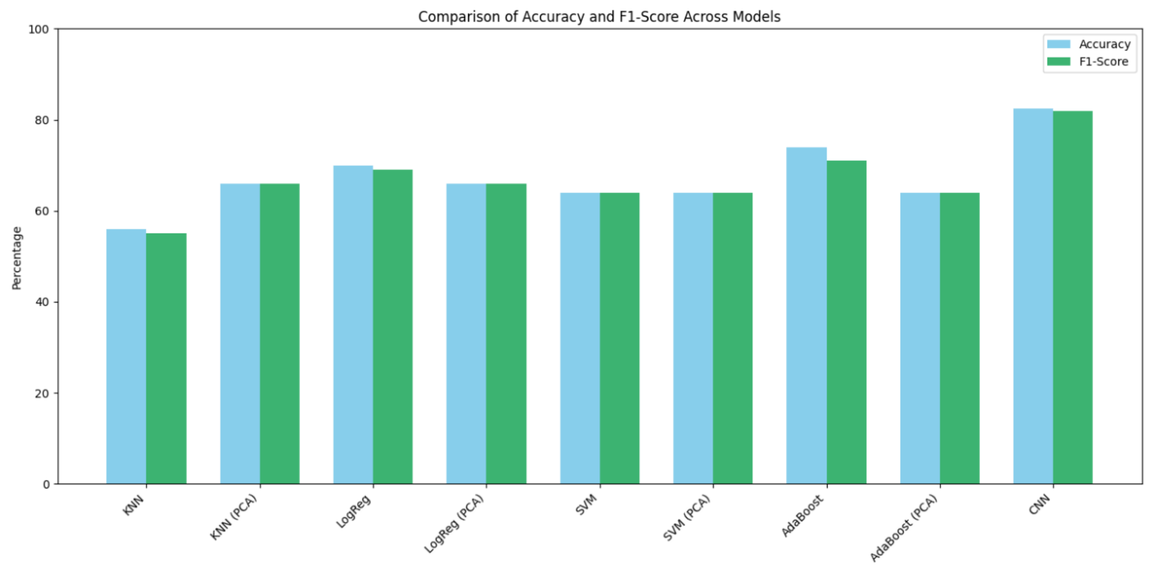


Figure Analysis

The bar chart compares Accuracy and F1-Score across all models. It shows that CNN significantly outperformed the other methods, achieving over 80% on both metrics. Simpler models like KNN and SVM had lower scores around 55–65%, while Logistic Regression and AdaBoost performed better, around 65–75%, with PCA sometimes improving results slightly but not consistently. This visualization highlights how deep learning was better at capturing subtle differences between real and AI-generated art, while traditional models struggled more with these fine distinctions.

6. Conclusions

The goal of this project was to develop models that can tell apart real artworks and AI-generated images. During the process, we discovered that simple models like KNN and SVM offered a useful starting point but struggled with reliably distinguishing subtle stylistic differences. More advanced models, including AdaBoost and Logistic Regression, performed somewhat better, while our CNN model achieved the best results, showing its ability to learn complex visual patterns directly from raw images.

We faced several key challenges along the way. One issue was that AI-generated images often closely mimicked real art styles, making them hard to separate even for humans. Our handcrafted features sometimes weren't expressive enough to capture these fine differences. Additionally, while our dataset was balanced in label count, the visual diversity between classes wasn't always even, complicating learning.

We also discovered practical challenges around training deep learning models. For example, we found that tuning the batch size had a noticeable effect on the CNN's convergence and final accuracy. Using too large or too small a batch sometimes led to unstable training or overfitting. Similarly, selecting the right learning rate and scheduler was critical—if it was too high, the model would diverge; too low, and training became very slow or stuck in poor local minima.

To address these challenges, we balanced the dataset classes carefully and experimented with feature extraction methods, such as color histograms and texture-based descriptors. We also applied PCA to reduce dimensionality, which helped in some models by removing noise and highlighting important variation. For hyperparameter tuning, we tried different settings and used early stopping to prevent overfitting.

For future work, we recommend improving feature engineering, potentially by using more advanced vision-language models like CLIP to get better representations of style and content. Expanding the dataset with more varied and higher-quality images would help models generalize better. We also suggest stronger data augmentation (random crops, rotations, color changes) to improve robustness, and more systematic hyperparameter searches (like grid or Bayesian optimization) to better tune learning rates, batch sizes, and architectures. Finally, managing randomness in cross-validation and ensuring reproducible experiments will be important for making these systems reliable and scalable.