Detecting Synthetic Human Faces Generated by StyleGAN2

Rashi Mahavir Solanki
Makers Lab, Tech Mahindra
rashisolanki2002@gmail.com

Veer Rajeev Sanghavi
Makers Lab, Tech Mahindra
veersanghavi8@gmail.com

***Abstract*—The proliferation of generative models like StyleGAN has necessitated robust methods for distinguishing AI-generated images from real ones. In this study, we introduce an advanced technique that combines StyleGAN2 inversion with comprehensive image similarity metrics to address this challenge. Our methodology involves extracting latent vectors from both real and AI-generated images through StyleGAN2 inversion and subsequently regenerating images from these vectors. We compute an array of image similarity metrics to quantify the resemblance between original and regenerated images, forming a detailed dataset of these metrics. This dataset is utilized to train sophisticated machine learning models, including Artificial Neural Networks (ANN) and XGBoost. The significance of this work lies in its potential to uphold the integrity of digital media and mitigate the spread of misinformation. Our experimental results demonstrate the high effectiveness of our approach, achieving an impressive 100% accuracy on our training and testing dataset in distinguishing between real and AI-generated images, highlighting its potential for real-world applications.**

***Index Terms*— StyleGAN2, Image Similarity, ANN, XGBoost**

I. Introduction

The rapid advancements in generative models, particularly StyleGAN, have revolutionized the field of image synthesis, enabling the creation of highly realistic artificial images. While these advancements have numerous positive applications, they also pose significant challenges, particularly in the realm of digital media integrity and the proliferation of misinformation. The ability to distinguish between AI-generated images and real images has become increasingly critical.

Existing methods for detecting AI-generated images primarily focus on direct analysis techniques, which often fall short in accuracy and robustness when faced with sophisticated generative models. In this context, we propose a novel approach that leverages the inversion of StyleGAN to extract latent vectors from images, both real and AI-generated. By regenerating images from these latent vectors, we can compute a comprehensive set of image similarity metrics that quantify the fidelity of the regenerated images relative to their originals.

By providing a reliable method to detect AI-generated images, this study contributes to efforts in maintaining the authenticity of visual content, thus helping to combat the spread of manipulated media and misinformation. The subsequent sections of this paper will elaborate on our methodology, experimental setup, and results. Our findings indicate a remarkable accuracy in distinguishing AI-generated images from real ones, underscoring the potential of our approach in real-world applications.

II. Data Description

This project focuses on distinguishing AI-generated images from real images, particularly facial images. The dataset comprises a total of 10,000 images, divided equally between AI-generated faces and real celebrity faces. The following sections provide detailed descriptions of the sources and characteristics of the images used in this study.

A. AI Generated Images

- **Source:** The AI-generated images are obtained from the website thispersondoesnotexist.com.
- **Generation Method:** These images are created using STYLE GAN 2, an advanced generative adversarial network (GAN) that synthesizes highly realistic human faces.
- **Number of Images:** 5,000
- **Image Characteristics:** Each image is a high-quality representation of a human face, typically devoid of common artefacts or distortions associated with less sophisticated GANs. These images are entirely synthetic and do not correspond to any real individuals.
- **Example Images:**



B. Real Images

- **Source:** The real images are sourced from the CineFace10 dataset on Kaggle.
- **Content:** This dataset contains facial images of celebrities, covering a diverse range of actors and actresses from various movies and television shows.
- **Number of Images:** 5,000

- **Image Characteristics:** The real images feature a variety of facial expressions, lighting conditions, and backgrounds typical of professional and candid photography. The celebrities' faces provide a rich set of real-world variations.
- **Example Images:**



III. Data Preprocessing

A. Format

Images are stored in a common format (e.g., JPG or PNG) suitable for processing by algorithms.

B. Annotations

Each image is labelled with its source category (AI generated or real) to facilitate supervised learning tasks.

This dataset provides a robust foundation for training and evaluating machine learning models aimed at distinguishing between AI-generated and real human faces. The inclusion of diverse and high-quality images from both categories ensures that the models can generalize well to different types of facial images.

IV. Methodology

This study focuses on the classification of AI-generated images and real-world photographs using advanced image analysis techniques and machine learning algorithms. The dataset was constructed by gathering AI-generated images from StyleGAN2, obtained from thispersondoesnotexist.com, and real images from the CineFace10 dataset on Kaggle, ensuring a diverse representation of real-world photographs. Initial preprocessing involved StyleGAN inversion to extract latent vectors from both sets of images, followed by image re-generation to create corresponding re-generated images. Image similarity metrics were computed between original images and their re-generated counterparts, including Euclidean Distance, Cosine Similarity (CLIP ViT), Mean Squared Error (MSE), Perceptual Loss, Structural Similarity Index Measure (SSIM), Peak Signal-to-Noise Ratio (PSNR), and Feature Extraction with SIFT.

The dataset was structured to include pairs of original and re-generated images, labelled as Class 0 (AI-generated and re-generated pairs) and Class 1 (real and re-generated pairs). Multiple machine learning and deep learning algorithms were explored and evaluated to select the optimal model for classification. Evaluation metrics such as accuracy, precision, recall, and F1 score were used to assess the performance of the models. The chosen model was then deployed to classify new images, ensuring robust performance in distinguishing AI-generated images from real photographs based on their similarity metrics. The methodology ensures a systematic approach to accurately classify and differentiate between AI-generated and real images using state-of-the-art techniques in image analysis and machine learning.

V. StyleGAN2 Architecture Overview

StyleGAN2 introduces advancements in generative adversarial networks (GANs), particularly in image synthesis and manipulation. Key components include a mapping network that transforms initial latent vectors into a disentangled intermediate space W, and a synthesis network that utilizes Adaptive Instance Normalization (AdaIN) and noise inputs to produce high-resolution, realistic images.

A. Mapping Network and Latent Spaces

1) Mapping Network:

The mapping network f transforms a simple input latent vector z (sampled from a standard normal distribution, Z) into an intermediate latent vector w. This network consists of multiple fully connected (FC) layers that help in disentangling the latent space. The mapping network aims to distribute the latent vectors more evenly in the latent space W, facilitating better control and manipulation of the generated images.

2) Latent Spaces Z and W:

The latent space Z is the initial space from which the input latent vectors are sampled. These vectors are then mapped to the intermediate latent space W through the mapping network. The latent space W is designed to be more disentangled, meaning that different aspects of the image (such as pose, texture, and color) are controlled more independently. This disentanglement enhances the ability to manipulate specific features of the generated images without affecting others.

3) Disentanglement:

Disentanglement refers to the separation of different generative factors within the latent space. In a well-disentangled latent space, changes in one dimension of the latent vector correspond to changes in a specific aspect of the generated image (e.g., changing the vector might alter only the pose or color of the image). This property is crucial for tasks such as image editing and style transfer, as it allows for precise control over the generated content.

B. Synthesis Network

1) Adaptive Instance Normalization (AdaIN):

AdaIN is a key innovation in StyleGAN2 that modulates the activations of the generator network based on the latent vector w. This modulation is achieved by adjusting the mean and variance of the activations, effectively controlling the style of the generated image at different levels of the network. AdaIN allows for fine-grained control over the

image synthesis process, enabling the generation of diverse and high-quality images.

2) Noise Inputs:
The synthesis network also incorporates noise inputs at various stages. These noise inputs add stochastic variation to the generated images, enhancing the realism by introducing minor variations and details.

3) Network Architecture:
The synthesis network g consists of multiple layers where each layer applies AdaIN and convolution operations. The network progressively up samples the spatial resolution of the image from a small starting resolution (e.g., 4x4) to the final output resolution (e.g., 1024x1024). This progressive growing approach helps in generating high-resolution images with fine details.

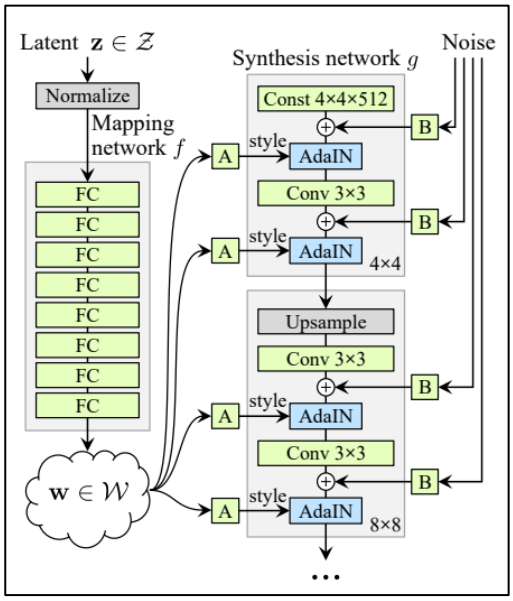


Fig. 1. StyleGAN2 Architecture

VI. High Level Solution Flow

A. Invert Image back to Latent Space

In this section, we explore the process of inverting an image to obtain its latent code using a pre-trained StyleGAN2 model. GAN inversion is crucial for understanding and navigating the latent space, allowing us to regenerate images and compare them to the originals for deepfake detection.

1) Image Preprocessing:
The target images are resized to 256x256 pixels, center-cropped, and normalized to match the input requirements of the StyleGAN2 model. This ensures that the images are in a consistent format for processing.

2) Latent Space Initialization:
Start with a random latent vector z from the latent space.

3) Optimization Strategy:
The optimization aims to find a latent vector w that, when passed through the generator, produces an image that closely resembles the target image. We use a perceptual loss function based on a pre-trained VGG network to measure the similarity between the generated and target images. The loss function also includes a noise regularization term to stabilize the optimization.

4) Iterative Optimization:
An Adam optimizer is used to iteratively update the latent vector and noise patterns. The learning rate is dynamically adjusted to fine-tune the generated image. Throughout the optimization, we monitor the progress by visualizing the generated image alongside the target image at regular intervals.

5) Final Output:
The optimized latent vector z is obtained, which is then used to regenerate the target image.

B. Generate image from the optimized Latent vector using StyleGAN Generator

Once the latent vector z has been optimized through the inversion process, we proceed to regenerate the image using the StyleGAN2 generator. StyleGAN2 incorporates a mapping network that disentangles initial latent vectors into an intermediate space W and a synthesis network that utilizes Adaptive Instance Normalization (AdaIN) and noise inputs to produce high-resolution, realistic images.

1) Input Latent Vector z:
The optimized latent vector z, obtained from the inversion process, serves as input to the StyleGAN2 generator.

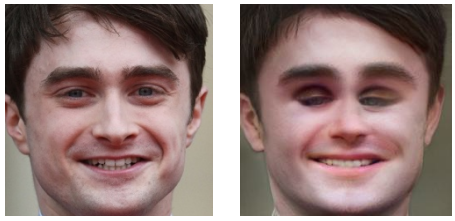
2) Image Synthesis:
StyleGAN's synthesis network g utilizes z to generate an image. This involves AdaIN for style control and noise inputs for stochastic variation.

3) Progressive Growing:
Starting from a low resolution (e.g., 4x4), the generator progressively up samples the image to the desired resolution (e.g., 1024x1024), maintaining fine details and high visual fidelity.

4) Final Output:
The output is a regenerated image based on the optimized latent vector z. This image is crucial for further analysis and comparison with the original target image.



a) AI-Generated Image *b) Re-generated Image*



a) Real Image *b) Re-generated Image*

Fig. 2. GAN Inversion Output

C. Perform Image Similarity on the re-generated and the input image

We conducted a rigorous evaluation of various metrics to quantify the similarity or dissimilarity between real and generated images. These include:

1) Euclidean Distance:

The Euclidean distance computes the straight-line distance between the two feature vectors.

a) Image conversion into 1- dimensional feature vectors:

Initially, the generated image is resized to match the dimensions of the real image for direct comparison. Subsequently, both images are flattened into 1-dimensional arrays. This transforms the images into feature vectors where each element represents a pixel value.

b) Formula:

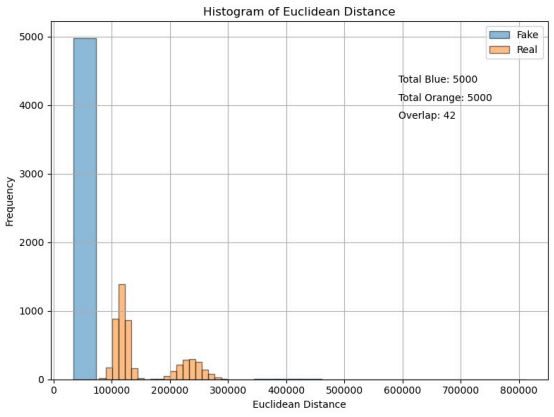
$d(a, b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$

where:

a and b are the flattened vectors representing real and generated images, respectively.

c) Resulting Score:

- A smaller Euclidean distance indicated greater similarity between the images, suggesting that they shared similar pixel patterns.
- A larger Euclidean distance indicates greater dissimilarity between the images, suggesting that the pixel values in the two images differ more significantly.



2) Cosine Similarity (Using CLIP Vision Transformer):

We employed the Vision Transformer (ViT) model from the CLIP (Contrastive Language-Image Pre-training) family, specifically the 'ViT-B-16-plus-240' model, pre-trained on the LAION-400M dataset, to compare the similarity between the two images. The embeddings of these images are compared using cosine similarity. Cosine similarity measures the cosine of the angle between two vectors, providing a metric of similarity.

a) Image Conversion and Preprocessing:

- Each image is converted to RGB format.
- The images are pre-processed using CLIP's preprocessing pipeline.

b) Image Encoding into a High-Dimensional Vector:

- The pre-processed images are passed through the CLIP model's image encoder.
- This process generates high-dimensional vector representations (embeddings) that capture the essential visual features of the images.

c) Formula:

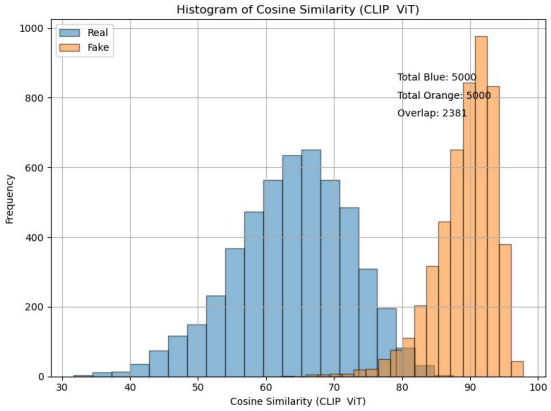
$$\text{Cosine similarity} = \cos(\theta) = \frac{A \cdot B}{\| A \| \, \| B \|}$$

where:

- A and B are the high-dimensional vectors(embeddings) of the two images.
- $A \cdot B$ is the dot product of the vectors.
- $\| A \|$ and $\| B \|$ are the magnitudes (norms) of the vectors.

d) Resulting Score:

- The cosine similarity score ranges from -1 to 1.
- The cosine similarity score is then scaled to a percentage value between 0 and 100 for easier interpretation by multiplying the cosine similarity score by 100.
- Scaled Similarity Score = $(\cos(\theta)) \cdot 100$
- This transformation provides a more intuitive percentage score, where:
  - 100% indicates the images are identical.
  - Lower scores indicate decreasing similarity.
  - A score of 0% would indicate no similarity.



3) Mean Squared Error:

MSE measures the average squared difference between corresponding pixels of two images, providing a quantitative assessment of image similarity.

a) Formula:

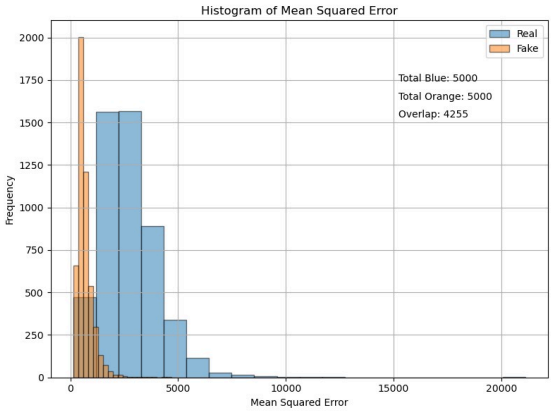
Mean Squared Error (MSE) = $\frac{1}{n}\sum_{i=1}^{n}(a_i - b_i)^2$

where:

a_i and b_i are the pixel values of the real and generated images, respectively.

b) Resulting Score:

- A lower MSE indicates higher similarity.
- A higher MSE indicates greater dissimilarity between the images.



4) Perceptual Loss Using VGG16:

Perceptual loss utilizes the VGG16 model to compare high-level perceptual features between images. The images are passed through the pre-trained VGG16 model to extract feature representations from specific layers.

a) Feature extraction:

The images are fed into the VGG16 model, and feature maps are extracted from chosen convolutional layers.

b) Loss Calculation:

The perceptual loss is computed as the difference between these feature maps.

c) Formula:

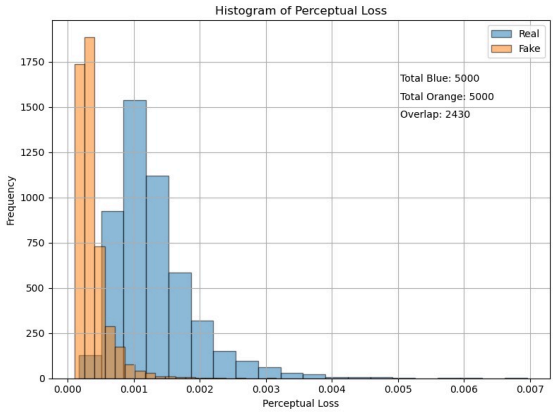
Perceptual Loss = $\sum_i ||\emptyset_i(a) - \emptyset_i(b)||^2$

where:

ϕ_i represents the feature maps from the i^{th} layer.

d) Resulting Score:

- A lower perceptual loss indicates higher perceptual similarity between the images.
- A higher perceptual loss indicates lower perceptual similarity between the images.



5) Structural Similarity Index Measure:

SSIM evaluates the similarity between two images based on luminance, contrast, and structure, providing a more perceptually relevant measure of image quality.

a) Formula:

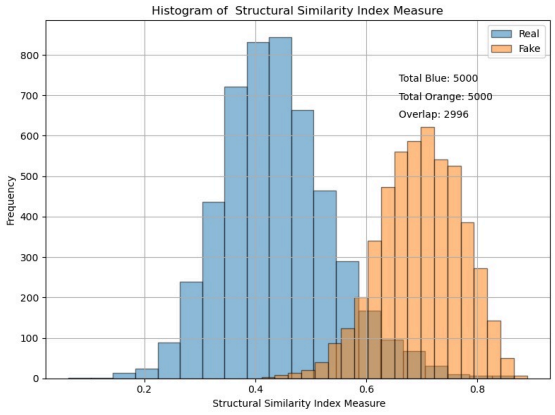
SSIM *(a, b)* = $\frac{(2\mu_a\mu_b+C_1)(2\sigma_{ab}+C_2)}{(\mu_a^2+\mu_b^2+C_1)(\sigma_a^2+\sigma_b^2+C_2)}$

where:

μ and σ denote the mean and standard deviation of the images, respectively, and C_1 and C_2 are constants.

b) Resulting Score:

- SSIM ranges from -1 to 1.
- 1 indicates perfect similarity.
- Values closer to -1 indicate dissimilarity.



6) Peak Signal-to-Noise Ratio:

PSNR quantifies the peak error between the original and generated images, providing a measure of image reconstruction quality.

a) Formula:

PSNR = $10 \cdot log_{10}(\frac{MAX^2}{MSE})$

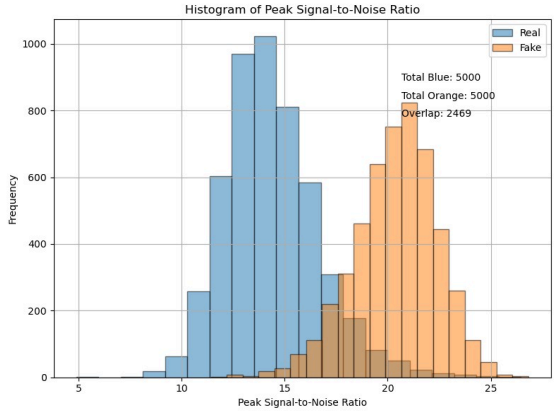
where:

MAX is the minimum possible pixel value and MSE is the mean squared error.

b) Resulting Score:

- Higher PSNR values indicate better image quality and higher similarity.

- Lower PSNR values indicate bad image quality and lower similarity.



7) Feature Extraction with SIFT (Scale-Invariant Feature Transform):

SIFT detects key points and descriptors in images to enable accurate image comparisons based on distinct features.

a) Key Point Detection:

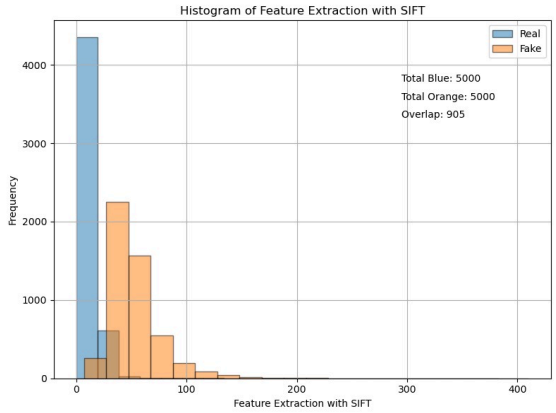
Identify key points in the images that are invariant to scale and rotation.

b) Descriptor Extraction:

Compute descriptors for each key point to characterize local image regions.

c) Resulting Score:

- The similarity between images is assessed by matching key points and comparing their descriptors.
- A higher number of matching key points indicates greater similarity.
- A lower number of matching key points indicates lower similarity.



➢ **Insights from Feature Importance Graph:**

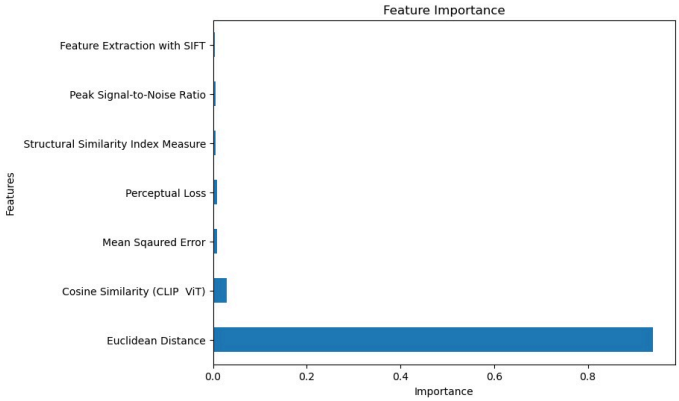


Fig. 3. Feature Importance Graph

We discovered that Euclidean distance and Cosine Similarity using CLIP Vision Transformer were the most important features. These metrics greatly helped us distinguish between re-generated and original images. They were particularly strong at capturing the overall meaning of images and the exact differences at a pixel level, which is crucial for accurately assessing how similar two images are.

D. Dataset Preparation for Model Training

For the dataset preparation, each real image is paired with its corresponding regenerated version obtained through StyleGAN inversion. Similarly, every generated (fake) image is paired with its regenerated counterpart. This pairing process ensures that each pair is aligned accurately for precise metric calculation, utilizing metrics such as Euclidean Distance, Cosine Similarity (Using CLIP ViT), Mean Squared Error, Perceptual Loss Using VGG16, Structural Similarity Index Measure (SSIM), Peak Signal-to-Noise Ratio (PSNR) and Feature Extraction with SIFT (Scale-Invariant Feature Transform). We then assign labels of real or fake to all the records. The inclusion of comprehensive image similarity metrics further enriches the dataset by providing quantitative measures between images and their regenerated versions.

1) *Splitting the Data for Training and Testing:* To effectively evaluate the model's performance, the data is split into training and testing sets. Using the "train_test_split" function from the sklearn library, 80% of the data (8,000 images) is allocated for training, while the remaining 20% (2,000 images) is reserved for testing.

2) *Feature Scaling:* To ensure that the model performs optimally, the features of the dataset are standardized. Standardization is a crucial preprocessing step that scales the features such that they have a mean of 0 and a standard deviation of 1. The "StandardScaler" is first fitted to the training data, which calculates the mean and standard deviation of the training set and then scales it accordingly. The same transformation is applied to the testing data to ensure consistency. The formula for standardization is given by:

$$X_{\text{scaled}} = (X - \mu)/\sigma$$

where:

X is the original feature, μ is the mean of the feature in the training data, and σ is the standard deviation of the feature in the training data.

E. Model Building

1) ANN

The neural network model is constructed with five layers, starting with an input layer and ending with an output layer. The first four layers consist of seven units each, utilizing the rectified linear unit (ReLU) activation function to introduce non-linearity. Following these hidden layers, the final layer incorporates a single unit with a sigmoid activation function, facilitating binary classification between real and generated images. The model is optimized using the Adam optimizer, chosen for its efficiency in handling large datasets and dynamic learning rates. Given the binary nature of the classification task, binary cross-entropy is employed as the loss function to measure the discrepancy between predicted and actual labels. Training parameters include a batch size of 32 instances per iteration and a total of 20 epochs to iteratively improve the model's performance and convergence. By the end of the training, the model achieved an accuracy of 100% with a corresponding loss of 0.0032, demonstrating its high effectiveness in distinguishing between real and AI-generated images.

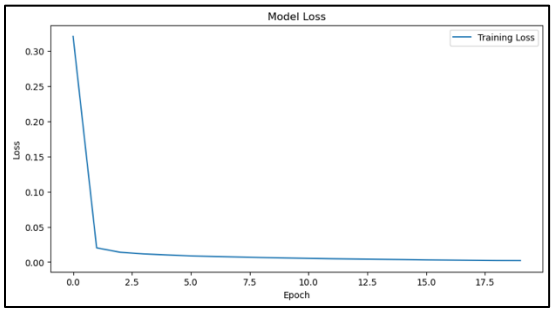


Fig. 4. Loss Graph

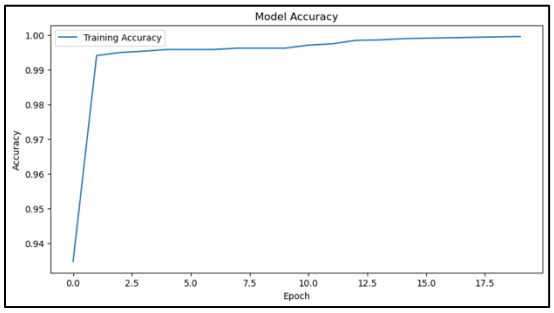


Fig. 5. Accuracy Graph

2) XG Boost

For the classification task, we utilize the XGBoost algorithm, known for its efficiency and superior performance in handling structured data. The XGBoost model is configured with the following hyperparameters: "colsample_bytree" is set to 1.0, indicating that all features are considered for each tree. The "gamma" parameter is set to 0.0, allowing for unrestricted tree partitioning. A learning_rate of 0.1 is chosen to balance the trade-off between convergence speed and model accuracy. The "max_depth" of each tree is limited to 3 to prevent overfitting while capturing sufficient data complexity. The "min_child_weight" is set to 3, ensuring that leaf nodes contain a minimum sum of instance weights to maintain model generalization. We use "n_estimators" set to 100, specifying the number of trees to build, which strikes a balance between model robustness and computational efficiency. The objective is defined as "binary:logistic", suitable for our binary classification task. Finally, a "subsample" rate of 0.8 is employed, which randomly samples 80% of the training data for each tree, further mitigating overfitting. Remarkably, the model achieved an accuracy of 100% on our testing dataset comprising 2000 images, demonstrating its exceptional capability in performing this classification task.

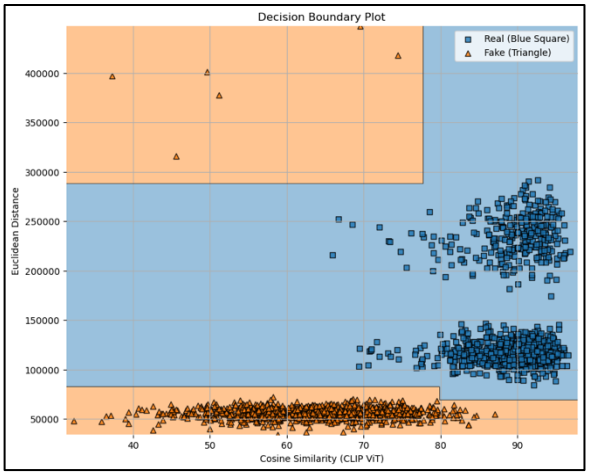


Fig. 6. 2D Decision Boundary Plot

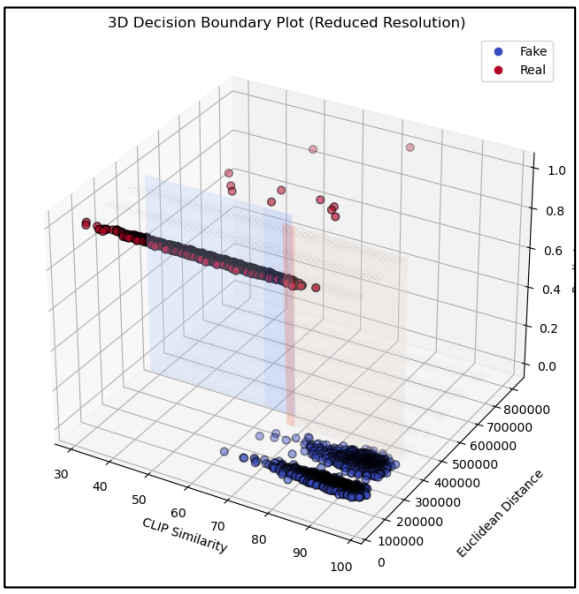


Fig. 7. 3D Decision Boundary Plot

3) KNN

For the task of distinguishing between AI-generated and real images, we employed a K-Nearest Neighbors (KNN) classifier, renowned for its simplicity and effectiveness in handling structured data. The KNN model was configured with a Manhattan distance metric, utilizing 11 neighbors for classification, and employing uniform weights across neighbors. This setup ensures each neighbor contributes equally to the decision-making process. Following training, the KNN model demonstrated exceptional performance, achieving a perfect accuracy rate of 100% on our testing dataset comprising 2000 images. This result underscores the robustness and reliability of using image similarity metrics in conjunction with KNN for accurately classifying AI-generated images.

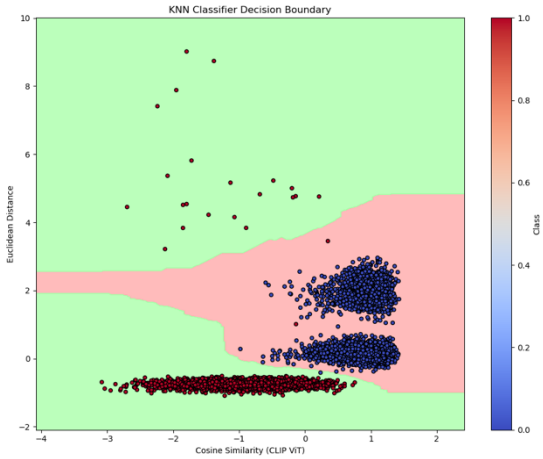


Fig. 8. KNN Classifier Decision Boundary Plot

F. Model Comparison and Selection

TABLE I
COMPARISON OF MODELS

	XGBoost	ANN	KNN	SVM	LR
Precision (Fake)	1.00	1.00	1.00	0.99	0.99
Precision (Real)	1.00	1.00	1.00	0.98	0.98
Recall (Fake)	1.00	1.00	1.00	0.99	0.98
Recall (Real)	1.00	1.00	1.00	0.99	0.99
F1-score (Fake)	1.00	1.00	1.00	0.99	0.99
F1-score (Real)	1.00	1.00	1.00	0.99	0.99
Accuracy	**1.00**	**1.00**	**1.00**	**0.99**	**0.99**

In selecting XGBoost over ANN and KNN, several factors prove pivotal. XGBoost excels in scalability, efficiently handling larger datasets and computation demands compared to KNN, which can become impractical with extensive data due to its storage requirements. Moreover, XGBoost offers clearer interpretability than ANN, making it easier to understand and explain model decisions, crucial for quality assurance and validation in complex projects like image classification. Despite achieving comparable 100% accuracy on testing data, XGBoost's computational efficiency further underscores its suitability for real-time applications, ensuring robust performance without compromising on speed or interpretative clarity.

G. Model Evaluation

Our neural network model, comprising five layers with rectified linear unit (ReLU) activations and a sigmoid output, achieved exceptional performance with 100% accuracy and a minimal loss of 0.0032. Trained over 20 epochs using the Adam optimizer and binary cross-entropy loss, the model demonstrated robust convergence and efficacy in binary classification. Visualizations of the training process showed steady improvements in both accuracy and loss metrics. Concurrently, the XGBoost algorithm, optimized with hyperparameters such as "colsample_bytree=1.0", "gamma=0.0", and "max_depth=3", achieved identical accuracy on a testing dataset of 2000 images. The model's confusion matrix further validated its reliability, with all predictions aligning perfectly with actual classifications. Comparative analysis highlighted XGBoost's superiority in scalability and interpretability over alternative models like KNN and neural networks, underscoring its suitability for real-time applications and complex image classification tasks.

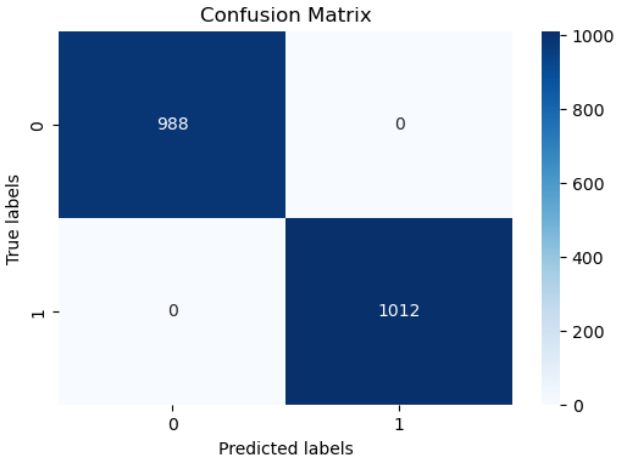


Fig. 9. Confusion Matrix

➢ **K-Fold Cross Validation**

We implemented a 10-fold cross-validation to rigorously evaluate the performance of our XGBoost model. This approach involved dividing the training dataset into 10 equal parts, training the model on 9 parts, and validating it on the remaining part, repeating the process 10 times. Our model achieved a mean accuracy of 99.91%, showcasing its high capability in distinguishing between real and AI-generated images. The standard deviation of 0.10% indicates the model's consistent performance across different data subsets, underscoring its reliability and generalisation capacity.

VII. Results

The performance of the XGBoost model was evaluated on a dataset comprising 10,000 images, equally divided between AI-generated faces (from StyleGAN2 via thispersondoesnotexist.com) and real faces (from the CineFace10 dataset on Kaggle). The model utilized Euclidean Distance and CLIP Similarity as features to classify the images. The Model achieved 100% Accuracy, Precision, Recall and F1-Score demonstrating high effectiveness.

This approach shows significant potential for real-world applications in digital media integrity and combating misinformation spread through AI-generated content.

VIII. Performance Analysis

The performance analysis of the GPUs used for inverting the StyleGAN and regenerating the image from the latent vector is a critical aspect of evaluating the efficiency and scalability of our methodology.

A. GPU Usage

Images are stored in a common format (e.g., JPG or PNG) suitable for processing by algorithms.

- Nvidia H100 80GB
- Nvidia A100 80GB
- Nvidia RTX 3060 Ti
- Google Colab T4 GPU

B. Inversion and Generation Time

- Nvidia H100 80GB: 25 seconds
- Nvidia A100 80GB: 40 seconds
- Nvidia RTX 3060 Ti: 60 seconds
- Google Colab T4 GPU: 130 seconds

The choice of GPU has a significant impact on the time efficiency of inverting the StyleGAN and regenerating images from latent vectors. The Nvidia H100 80GB provides the best performance, making it the most suitable for large-scale and real-time applications. The Nvidia A100 80GB offers a good balance of performance and cost-efficiency, while the Nvidia RTX 3060 Ti is a viable option for more budget-conscious projects. The Google Colab T4 GPU, despite its longer processing times, remains a valuable resource for those with limited access to advanced hardware.

IX. Conclusion

This study addresses the critical need for effective detection of AI-generated images in an era where generative models like StyleGAN are increasingly sophisticated. By developing an innovative approach that combines StyleGAN inversion with a detailed analysis of image similarity metrics, we have demonstrated a highly accurate method for distinguishing between real and AI-generated images. Our experiments show that the computed image similarity metrics form a comprehensive dataset, enabling the training of robust machine learning models. The use of XgBoost over the ANN for it's less computationally taxing and better interpretability has proven to be highly effective. These results highlight the potential of our approach to be implemented in real-world applications, providing a reliable tool for maintaining the authenticity of digital media. Through this research, we not only contribute a new technique to the field of AI-generated image detection but also lay the groundwork for future advancements. The implications of this study extend to various domains, including cybersecurity, media verification, and beyond, demonstrating the far-reaching impact of our findings.

X. Future Work

For future work, we hypothesize that our methodology can be expanded to detect AI-generated images from various types of GANs by developing multiple end-to-end systems tailored to each GAN architecture. By constructing a battery of powerful GANs, such as StyleGAN2, BigGAN, CycleGAN, and SAGAN, each capable of inverting and regenerating images specific to their design, we can create distinct sets of image similarity metrics for each GAN type. These individual systems will be trained separately to recognize patterns unique to their corresponding GANs, allowing for precise detection of AI-generated images from a wide range of GAN architectures. This approach would significantly enhance the robustness and versatility of our detection capabilities, ensuring accurate identification of AI-generated images regardless of the specific GAN used in their creation.