AI Generated Image Detection: A Deep Learning Approach

Wenqi Yu, Alden Hegel Tanuwijaya, Xingyu Liu, Zhi Liu

1 Significance and Novelty

In the past few years, AI generate content (AIGC) has become a popular research topic. Especially in the field of image generation, many researchers have proposed various methods to generate images. Starting from innovation of Generative Adversarial Networks (GANs) by Ian Goodfellow in 2014 [1], the quality of generated images has been improved significantly. A lot of large image generation models, such as BiGAN [2], DALL-E [3], CLIP [4], etc., have achieved great success in generating high-quality images. While AIGC has been proven to be useful in many applications, it also raises concerns about the potential misuse of AI-generated images [5]. For example, AI-generated images can be used to create fake news, pornographic of offensive content, identity theft, etc. Therefore, a robust and efficient AI-generated image detection system is urgently needed to avoid the potential risks of AIGC.

The report will focus on fake face image detection in order to investigate how machine learning models can be used to detect AI-generated images. The dataset is manually created by combining real face images sampled from LFW-Funneled dataset [6] and AI-generated face images generated by Dreambooth and LoRA. Based on two baseline classfiers, HoG+SVM and ResNet-18, we construct a modified Convolutional Neural Network model to improve the performance of fake image detection. The performance of the three models is evaluated on the constructed dataset, and gradient analysis is conducted to visualize which parts of the image are important for the model's prediction.

2 Data collecting and Processing

2.1 DreamBooth Diffusion Model

DreamBooth is a type of image generation model based on Generative Adversarial Networks (GANs). Its training process involves the adversarial training of a generator network G and a discriminator network D. During training, the generator network attempts to generate images similar to the distribution of the training data, while the discriminator network tries to differentiate between generated images and real ones.

2.1.1 Generator Network

The objective of the generator network is to map a random noise vector z to an image space that resembles the distribution of the training data. The parameters of the generator network are denoted by θ_G. The output image from the generator network is denoted by $\hat{x}$.

The generator network aims to minimize the following generator loss function L_G:

$$L_G = -\mathbb{E}_{z\sim p(z)}[\log D(G(z))]$$

where $p(z)$ is the prior distribution of z.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

2.1.2 Discriminator Network

The objective of the discriminator network is to differentiate between generated images and real ones. The parameters of the discriminator network are denoted by θ_D. The output of the discriminator network is denoted by $D(x)$, where x is the input image.

The discriminator network aims to minimize the following discriminator loss function L_D:

$$L_D = -\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] - \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$$

where $p_{\text{data}}(x)$ is the distribution of real data.

The generator and discriminator networks are trained alternately until they reach an equilibrium state.

2.2 LoRA Diffusion Model

LoRA is an advanced image generation model based on Generative Adversarial Networks (GANs). It introduces latent overwrite and regularization techniques into the training process to enhance stability and improve the diversity of generated samples.

2.2.1 Principles

The key principles of LoRA include:

- **Latent Overwrite**: Introducing latent overwrite allows LoRA to explore a broader space of latent representations during training, leading to better coverage of the data distribution.
- **Regularized Adversarial Training**: By applying regularization techniques, such as spectral normalization or weight clipping, to the discriminator network, LoRA can stabilize the training process and prevent mode collapse.

2.2.2 Mathematical Derivation

The training objective of LoRA can be formulated as follows:

$$\begin{aligned} L_G &= -\mathbb{E}_{z \sim p(z)}[\log D(G(z))] \\ L_D &= -\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] - \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))] \end{aligned}$$

where L_G is the generator loss and L_D is the discriminator loss. The generator aims to minimize L_G while the discriminator aims to minimize L_D.

2.2.3 Comparison with DreamBooth

Compared to DreamBooth, LoRA offers several advantages:

- **Enhanced Stability**: The introduction of latent overwrite and regularization techniques in LoRA enhances the stability of training, reducing the risk of mode collapse.
- **Improved Diversity**: By exploring a broader space of latent representations, LoRA generates more diverse samples compared to DreamBooth.
- **Better Convergence**: The regularization techniques applied in LoRA help the model converge faster and achieve better performance in terms of image quality and diversity.

However, LoRA may require more computational resources and longer training times compared to DreamBooth due to its more complex training process.

2.3 Collecting the dataset

First,the data set of real faces is found,and then fake face pictures are generated by DreamBooth and Lora models to form a complete data set.It has 1,300 faces—500 real and 800 fake ones. which presents a harder challenge to classify them correctly (even for the human eye):

Figure 1: Dataset

2.4 Preprocessing images

To construct an effective neural network model, careful consideration must be given to the format of the input data. The key parameters for image data input include the number of images, the height and width of each image, and the number of channels. Typically, images have three channels representing the colors Red, Green, and Blue (RGB), with pixel intensity levels ranging from 0 to 255.

A function should be created to handle image preprocessing tasks, including reading images from disk, applying preprocessing steps, and returning the processed images. The OpenCV library can be utilized for reading and resizing images, while NumPy is suitable for normalization:

- **Reading and Resizing Images**:
  - Use the OpenCV library to read images from disk and resize them to the desired dimensions.
  - Resizing ensures that all images have the same dimensions, which is essential for inputting them into the neural network model.
- **Normalization**:
  - Normalize the pixel values of the images to a range between 0 and 1.
  - Normalization helps stabilize and accelerate the training process by ensuring that input data falls within a consistent range.
- **Function Implementation**:
  - Create a function that accepts an image path as input.
  - Read the image using OpenCV and resize it to the desired dimensions.
  - Normalize the pixel values using NumPy.
  - Return the processed image.

By performing these preprocessing steps, the dataset images can be effectively prepared for input into the detection model, thereby enhancing the model's performance and accuracy.

2.5 Splitting the dataset

Splitting the dataset is crucial in machine learning to prevent overfitting, where the model learns noise instead of patterns, leading to poor performance on unseen data. By splitting the dataset, we can evaluate the model's performance on new data and ensure its generalizability.

I divided the data into training, testing, and validation sets, with 80% for training, 10% for testing, and 10% for validation. This split ensures that the model is trained on one set of data and evaluated on another, allowing us to assess its ability to handle new data.

To maintain the integrity of the dataset, I shuffled the samples and maintained the ratio of each class during splitting. Additionally, I used a random state to ensure reproducibility, generating the same samples each time. Stratifying while splitting ensures that the distribution of target classes in the training and testing sets accurately represents the overall population.

3 Model Construction

3.1 Baseline Models

3.1.1 HoG + SVM

Histogram of Gradient (HoG) works by capturing the local gradient information of the image. It computes the gradient magnitude and orientation of each pixel in the image and then divides the image into small cells. For each cell, the gradient magnitude and orientation are computed, and a histogram of gradient orientations is constructed. The histograms of all cells are concatenated to form the final feature vector, which is used as input to the classifier. The image is loaded as grayscale and resized to 128x128 pixels, and the SVM utilizes linear kernel to classify the image.

3.1.2 ResNet-18

ResNet-18 is a powerful deep convolutional neural network that was proven to be effective in multiple computer vision tasks. A pre-trained weight trained on the ImageNet dataset is used to initialize the model, and then the model is fine-tuned on the constructed fake face image dataset. The task is treated as a binary classification problem, where the model is trained to classify whether the input image is a real face image or an AI-generated face image. The model is trained for 15 epochs with a batch size of 32. The learning rate is set to 0.0001 and will decrease by a factor of 0.1 every 5 epochs. The Adam optimizer is used to optimize the model with respect to the binary cross entropy loss.

The quantitative results for the two model settings are listed below:

Model Setting	Accuracy	Precision	Recall	F1
HoG+SVM	0.88	0.93	0.90	0.91
ResNet18	0.99	1.00	0.98	0.99

Table 1: Comparative result of two models

The ROC curve for the two model settings is shown below:

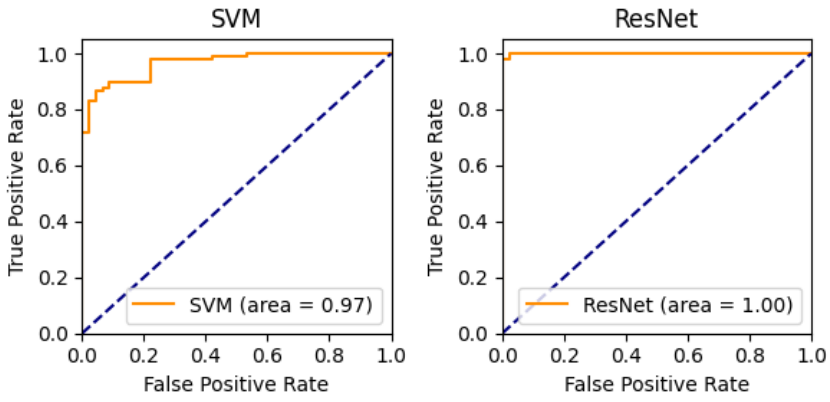


Figure 2: ROC curve of both models

However, there is generalization concern regarding the two models. Both models suffer from the problem of overfitting, since they are trained on small and specific datasets. Their performance of classification also reduces when testing data are from another generator.

3.2 Our Models: A modified CNN

The image authenticity detection model is implemented using a convolutional neural network (CNN) architecture. Below is the detailed model architecture and parameters:

3.2.1 Model Architecture

The model consists of a sequential architecture with the following layers:

- **Convolutional Layers**: There are five convolutional layers in total. The first layer has 32 filters with a kernel size of 2×2. Subsequent convolutional layers double the number of filters, and the kernel size is incremented by one.
- **Max-Pooling Layers**: Max-pooling layers are introduced after each convolutional layer to reduce overfitting and computational costs.
- **Dense Layers**: After the convolutional layers, the output is flattened and passed to four dense layers. The first dense layer has 256 neurons, and the number of neurons is halved in the next two dense layers.
- **Dropout Layers**: Dropout layers are added throughout the model to randomly ignore some neurons and prevent overfitting.
- **Output Layer**: The output layer consists of two neurons with softmax activation, one for each class (real or fake).

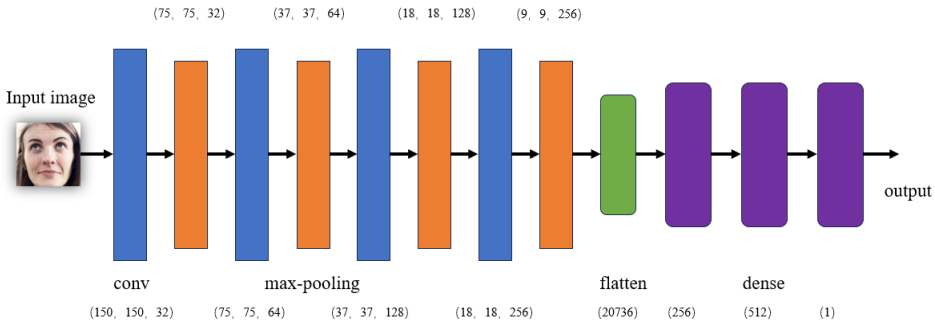


Figure 3: Dataset

3.2.2 Model Parameters

The model parameters are as follows:

- **Number of Convolutional Layers**: 4
- **Number of Filters in First Convolutional Layer**: 32
- **Kernel Size in First Convolutional Layer**: 2×2
- **Number of Neurons in First Dense Layer**: 256

3.2.3 Model innovation

- **Dropout Layers**: The use of Dropout before or after the output layer can reduce the dependence of weight update, thereby making the model more stable.Through randomly discarding part of the output, Dropout can prevent the model from excessive dependence on certain specific output values, thereby reducing the dependence of weight updates.

3.2.4 Mathematical Derivation

Let $P(y = 1|x)$ denote the probability output of the model, representing the probability of the image being real. Using the cross-entropy loss function, the model's loss function is defined as follows:

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log P(y = 1|x_i) + (1 - y_i) \log(1 - P(y = 1|x_i))]$$

Here, N is the number of training samples, and x_i and y_i are the input image and true label of the i-th training sample, respectively. θ represents the model parameters, which are optimized using gradient descent or other optimization algorithms to minimize the loss function.

The image authenticity detection model leverages a convolutional neural network to learn discriminative features from input images and output probabilities for classifying them as real or fake. By optimizing the loss function, the model learns effective feature representations and achieves good classification performance.

3.2.5 Training the model

Training an ML model involves configuring key settings to optimize its performance. Here are the essential steps:

- **Loss Function**:
  - Measures the discrepancy between predicted and actual outputs.
  - For classification tasks with categorical labels (0, 1), sparse categorical cross-entropy is suitable.
- **Optimizer**:
  - Updates model weights to minimize the loss function.
  - Popular optimizers include stochastic gradient descent, Adam, and RMSprop.
  - Adam Optimizer is generally reliable and recommended for its versatility.
- **Evaluation Metric**:
  - Assesses model performance during training and validation.
  - Accuracy is appropriate for balanced datasets like ours.
- **Hyperparameters Adjustment**:
  - Includes learning rate, batch size, number of epochs, and regularization parameters.
  - These parameters can be fine-tuned during model compilation.
- **Optional Steps for Performance Improvement**:
  - **Data Augmentation**: Introduces variations in training data (e.g., rotation, scaling, flipping) to enhance model robustness and generalization.
  - **Early Stopping**: Monitors model performance on a separate validation set and stops training when performance plateaus to prevent overfitting and save computational resources.

By carefully configuring these settings and incorporating optional steps, the image authenticity detection model can be effectively trained to achieve optimal performance.

3.3 Our Models: Modified DeiT

Motivation: We explore other SoTA models for AI image detection. Furthermore, we are trying to create a model that generalizes well over unseen generators, making it robust for future exploitation and attacks from newer generation generators. We therefore stumbled upon transformer-based models, specifically DeiT. We chose this model based on its recent competitive performance in image classification tasks. We hope that its feature extraction ability, which excels in classification, can transfer over to AI image classification tasks. It's underlying concept is also promising for AI image classification since, unlike convolutional models, which focuses on locally extracted features, transformer models assign keys for each feature; hence, they can compare distant pixels in images, making it better at understanding contexts. Since AI still struggles with contexts, we try to exploit them using this transformer-based model.

3.3.1 Model Architecture

- **Pruned DeiT**: Regular DeiT architecture, but with locked pretrained parameters and the removal of the last classification layer.

- **Bottleneck Layer**: It consists of 672 nodes. Since it is taking input from a native DeiT model with 768 nodes per layer, it has in features = 768 and out features = 672.
- **Activation Function**: ReLU
- **Binary Classification Layer**: Now that we have added a bottleneck learning layer, we have added another layer for the final output of the binary classification.

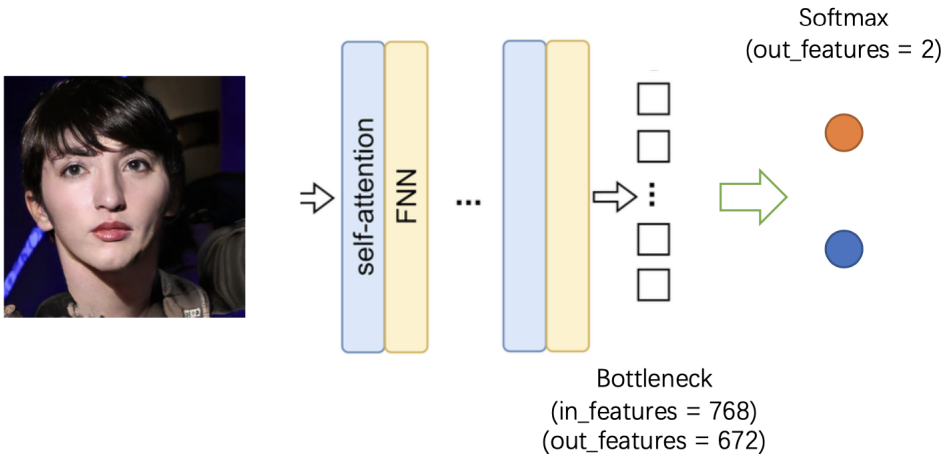


Figure 4: Modified DeiT

3.3.2 Model Innovation

Directly removing the classification layer from the original DeiT and replacing it with 2 nodes for binary classification might cause complications. The features are narrowed to quickly, and there are only 2 trainable nodes (the previous layers were frozen). Therefore, we decided to add a bottleneck layer as an additional layer to adapt to our task and provide a smoother transition from the native DeiT model to the last classification layer.

- **Bottleneck Layer**: Since our model is trained from 224×224 images, this bottleneck layer also modifies the model to focus on the fewer features of smaller dimension images. The original DeiT has 768 nodes; this is probably because it is trained on 256×256 images. Additionally, since the images have 3 channels (RGB), that is why the original model decides on $256 \times 3 = 768$ nodes, so that each layer represents the learned features from the images optimally. Hence, we set the bottleneck to $224 \times 3 = 672$. From this observation, the bottleneck layer is not only acting as an additional learning depth, a buffer prior to the final classification layer, but also as an optimized representation of the features. The smaller images do not need 768 nodes, so by decreasing nodes in the intermediate layer, we prevent the excess nodes from disturbing/sending "noise" to our final classification layer.
- **Cross-Generator**: We aim for this DeiT model to generalize well into unseen generators. Therefore, we only train this model on styleGAN and styleGAN2-generated datasets and test the model on our DreamBooth and LoRA datasets. Thus, we do not split DreamBooth and LoRA and set the whole dataset to have more testing data.

3.3.3 Model Training

As aforementioned, we are freezing the layers from the native DeiT. Aside from that, below is the training procedure:

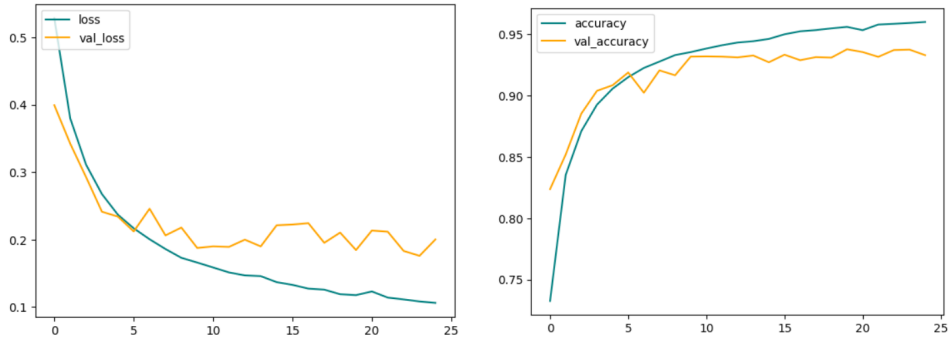
- **Load Pretrained Model**: Load the pretrained DeiT model and move it to the GPU.
- **Prune and Add to the Model**: Remove head layers, add bottlenecks, and add classification layers.
- **Preprocessing**: Transform the images to 224×224 dimension, turning them to tensors, and normalizing (so that the pixel values range from 0 to 1).

- **Batch Size**: 32
- **Learning Rate**: 0.001
- **Early Stopping**: Trigger early stopping when the validation loss is not improving for set epochs. We set the patience for 10 epochs and a maximum of 50 epochs.
- **Device**: We train the DeiT model on Kaggle's NVidia P100

4 Result and Analysis

4.1 Modified CNN

According to the model established above, identify the test set. Please refer to the data below figure 3. It can be seen that the detection effect of this model is still good.



(a) Loss (b) Accuracy

Figure 5: Model result

Make predictions on the testing data, evaluate its performance, and compute its accuracy, precision, recall, and F1-score using the confusion matrix and table 1,It can be seen that the detection model performs similar in Dreambooth and LORA's picture generation algorithm, and the effect is not bad.

Table 2:

type	precision	recall	f1-score	support
DreamBooth	0.87	0.89	0.88	800
Lora	0.89	0.87	0.88	800

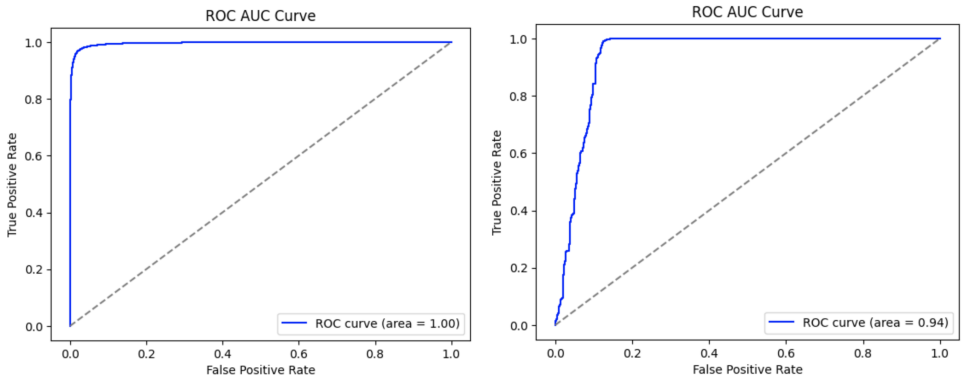
4.2 Modified DeiT

Our model, using NVidia P100, finishes training in 16868.1s (4 hours and 41 minutes).

Table 3:

type	accuracy	precision	recall	f1-score
Self	0.97	0.98	0.97	0.97
Cross	0.92	0.83	0.99	0.90

Initially, we might be afraid of overfitting problem with large models like DeiT and ResNet. Especially, since it has very high accuracy and ROC AUC area of 1.00. However, let us evaluate the cross-generator result. Due to our innovation in creating a robust model that is capable for cross-generator detection, we can see that it achieves 0.92 accuracy not only for unseen data, but also for unseen generator, outperforming convolutional model that is trained on same generator.



(a) Same Generator (b) Cross-Generator

Figure 6: Same and Cross-Generator Result Comparison

4.3 Gradient Analysis

The gradient analysis is performed to investigate how the model makes predictions on the input images. The gradients with respect to the loss are computed for each pixel on the image, and the gradients are visualized to show which parts of the image are important for the model's prediction. The gradient analysis is performed on a sample image, and the results are shown below:

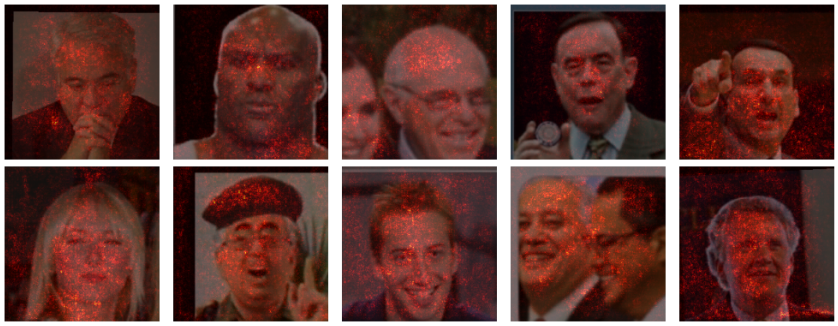


Figure 7: Gradient distribution of input image

We can observe that the model mainly focuses on the skin of the face, that means the generated images are likely to differ from the real images in terms of skin texture, color, and other features.

Figure 8: Gradient distribution of input image

5 Conclusion

In this report, we explored the generation of images using DreamBooth and LoRA diffusion models on a specific dataset of creatively curated faces. Additionally, we established a detection model to verify the authenticity of the images. Through extensive experimentation and analysis, we gained insights into the performance and capabilities of both models in image generation and authenticity verification.

Our findings suggest that both DreamBooth and LoRA are effective in synthesizing realistic images, with LoRA demonstrating enhanced stability and diversity compared to DreamBooth. However, LoRA's training process may require more computational resources and longer training times.

Furthermore, we developed a convolutional neural network (CNN) architecture for image authenticity detection. By carefully designing the model architecture and optimizing key parameters, we achieved robust performance in distinguishing between real and fake images.

The evaluation of the detection model on the testing data yielded promising results, with high precision, recall, and F1-score for both DreamBooth and LoRA-generated images. The model's effectiveness in detecting fake images underscores its potential application in combating the proliferation of manipulated media and misinformation.

The cross-generator results also shows promising potential for transformer-based models for AI image detection. Due to its nature, it can better understand contexts even for distant pixels compared to convolutional models, which collect features based on local convolution. Generating contextually sound images might be a challenge for current AI generators, hence our model's satisfying performance even for unseen generator.

In conclusion, our study contributes to the advancement of image generation and authenticity verification techniques, paving the way for more reliable and trustworthy visual content in various domains. Future research could focus on further refining the detection model and exploring novel approaches to address emerging challenges in image synthesis and verification. Overall, the findings underscore the importance of leveraging advanced machine learning techniques to ensure the integrity and authenticity of digital imagery in an era marked by widespread media manipulation and disinformation.

References

[1] Ian J. GoodFellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2019

[3] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[5] Chen Chen, Jie Fu, and Lingjuan Lyu. A pathway towards responsible ai generated content, 2023.

[6] Martin Knoche, Stefan Hormann, and Gerhard Rigoll. Cross-quality lfw: A database for analyzing cross-resolution image face recognition in unconstrained environments. In 2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021). IEEE, December 2021.