

# Detecting GAN Generated Images

## Current Techniques and Future Directions

Rakesh Santhakumaran  
Indiana University  
Bloomington  
rsantha@iu.edu

Kamalesh Kumar MG  
Indiana University  
Bloomington  
kmandak@iu.edu

Harini Sugumar  
Indiana University  
Bloomington  
hasugumk@iu.edu

Aravind Sivakumar  
Indiana University  
Bloomington  
arsivak@iu.edu

### Abstract

*Generative adversarial Networks (GANs) has become prevalent in producing realistic images, videos and other forms of data for malicious purpose such as spreading fake news in social media. In order to detect and stop the exploitation of this technology, it is now crucial to detect GAN generated images. GAN generated images often have some artifacts different from real images. So, In this project, we explored four different approaches for detecting GAN generated images: Analysing by exploiting inherent differences in the co-occurrence patterns of different image bands, a Fast Fourier Transform (FFT) is applied on each of the RGB channels of the input images and then deep learning technique applied, analyzing frequency space to identify artifacts, and using a support vector machine (SVM) classifier to distinguish between real and fake images using the landmark location of facial features. We used dataset of GAN generated images provided by Professor Fil Menzer and a dataset of real images from CELEB-A HQ which contains images of celebrities. The performance of all approaches was evaluated on the testing dataset and some real world images from Instagram public profiles. Our models were found to be effective in detecting style GAN generated images. In the future, we aim to generalize our approaches to detect GAN generated images by other architecture.*

**GitHub Link:** <https://github.com/rsantha-kmandak-arsivak-hasugum-finalProject/tree/main>

### 1. Introduction

Generative Adversarial Networks also called as GANs are capable of producing realistic images, videos and other sort of data. GANs contain two neural networks: generator and a discriminator which work together in generating data that are identical to real ones. The generator creates bogus data samples and the discriminator attempts to find the difference between the fake and actual samples. These GANs are incredibly good at convincing people that they

are real. For example, whichfaceisreal.com (West & Bergstrom, 2019), a website where a user can view two different images, one from Flickr faces HQ data set and one generated by style GAN and identify which of this image is real for them. These fake GAN images often deceive humans and provide difficulty to distinguish them from the real images.

Although GANs can be used for many good applications, they can be easily used for malicious purpose such as making convincingly fake images or videos that are hard to spot. Recently there has been an increase in the fake accounts and mis information spread in social media using GAN synthesized images. This has created a practical problem of misleading and influencing public in internet world. So detecting GAN generated images is now a crucial task to address the potential misuse of these generative models.

Several research has been conducted for the purpose of identifying Gan generated images. Analyzing the statistical characteristics of these images has been a popular technique in detecting GAN generated images. As the images produced using GAN generally have different statistical characteristics from real images. For example, images produced by GAN have different facial features, color distribution, texture patterns and frequency components. So, identifying these discrepancies provide a way to distinguish these GAN generated images from real ones. Several machine learning techniques including Support Vector Machine (SVM), Random Forests, Convolutional Neural Network (CNNs) have been used in classifying real and GAN generated images with high accuracy. In this report, we have provided a review of current state of art methods in detecting GAN generated images including their advantages and limitations. In addition, we have also discussed about four different approaches we tried for detecting Gan generated images and have provided the results and observation. Ultimately, the goal is to provide overview of the current state of research in detecting GAN generated images and to help guide future research in this area.

## 2. Background

The rapid raise of deep learning and GANs have made it easier than ever to generate high quality images and videos close to realism. Due to the growing concerns about the possible use of GANs for malicious purposes, there has been a recent surge in research on identifying GAN generated images. Several techniques were suggested for detecting GAN generated images. One such approach was detecting GAN generated faces by identifying irregularities in corneal specular highlights proposed by Hu et al. [1]. In this research, it was demonstrated that the specular highlights in GAN generated images frequently differ from the direction of the light source and may be used to spot fake images. Based on cross band cooccurrences analysis, Barni et al. [2] presented a convolutional neural network (CNN) to detect GAN generated facial images. The cross band cooccurrence matrix (CCM) of actual and fake facial images are examined by the author and significant differences between them were found. Later, to recognize fake images, a CNN model was used to train these attributes. Frank et al. [3] suggested a frequency analysis for identifying deep fake images. The real and GAN images were analyzed in frequency space to identify unique artifacts in high frequency range. Based on these artifacts, a random forest classifier was used to detect fake images. Another different approach of identifying GAN synthesized faces based on landmark analysis of facial feature positions was proposed by Yang et. Al. [4]. The author identified notable changes between real and the GAN image landmark locations. However, the accuracy and robustness of detecting methods still need improvement and, in this project, we adapted and implemented these methods and evaluated their performance on dataset of GAN generated images and real images from CELEB-A HQ dataset. By comparing and analyzing the results, we were able to identify advantages and disadvantages of each technique and were able to suggest future directions for improving GAN detection.

## 3. Data

### 3.1. GAN Generated Images

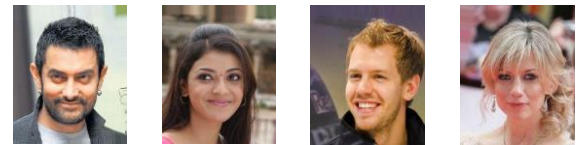
For GAN generated images, we were provided a google drive link which contained three folders psi-1.0, psi-0.7 and psi-0.5. Each folder contained 100 sub folder each of which contains 10,000 images with a resolution of 1024x1024 pixels. There were total of 3,00,000 images, out of which we choose 10,000 images for training purposes in all four of our approaches.



**Fig. 1:** Examples of GAN synthesised images of realistic human

### 3.2. Real Images

For real images, we used CelebA dataset which is now a well-liked dataset for training machine learning and deep learning models for variety of face related tasks. It has over 2,02,599 images of celebrity with resolution of 1024x1024 pixels. The images in CelebA are centered and cropped such that faces are around the same size and orientation. For training purpose of our four approaches we choose 10,000 images from the CelebA dataset.



**Fig. 2:** Examples of Real images from Celeb A dataset

## 4. Methods

### 4.1. Eye Coordinate Analysis

In this method, we investigate the eye coordinates of the GAN generated images to identify inconsistency. The lack of physical limitations in the GAN models is what causing this discrepancy. The eye positions of GAN generated images are in same position.

#### 4.1.1 Experiment Setup

This approach detects the facial landmarks, specifically the positions of the eyes in a given input image. We first loaded the face detector and landmark predictor provided by the Dlib library. Then, we loaded the input image and converted it to grayscale. We then used the face detector to detect faces in the grayscale image. For each detected face, we used the landmark predictor to identify the locations of the left and right eye corners. We then calculated the center of each eye by taking the average of the coordinates of the left and right eye corners. To visualize the results, we plotted circles on the detected eye centers using the OpenCV library. Finally, we displayed the output image and saved it to a file. Overall, our methodology involved loading the necessary libraries, loading the input image,

detecting the facial landmarks, and calculating the eye centers.

## 4.2. RGB Analysis

### 4.2.1 Using cross band co-occurrences

This approach can effectively differentiate between GAN-generated images and real images by exploiting the inherent differences in the co-occurrence patterns of different image bands by analyzing the cross-band co-occurrences in GAN-generated face images.

#### 4.2.1.1 Experimental Setup

**i. Data Preprocessing:** The code defines a set of functions to compute co-occurrence matrices for an input image. The function `compute spatial cooccurrence` takes a single channel of an image and a delta value as inputs, where delta is a tuple of two integers representing the displacement along the x and y axes. It then computes the co-occurrence matrix for the given channel by iterating over each pixel and incrementing the corresponding entry in the matrix based on the values of neighboring pixels within the given displacement. The resulting matrix is returned. The function `compute crossband cooccurrence` takes an RGB image and two delta values as inputs. It computes the cross-band co-occurrence matrix for a pair of channels (R-G, R-B, or G-B) by iterating over each pixel in the R channel and incrementing the corresponding entry in the matrix based on the values of neighboring pixels within the given displacements. The resulting matrix is returned. The function `compute cooccurrence matrices` takes an RGB image as input and computes all six co-occurrence matrices by calling the `compute spatial cooccurrence` and `compute crossband cooccurrence` functions. The resulting matrices are stacked along the last axis to create a tensor of shape (256, 256, 6). Finally, the function `preprocess image` takes an image as input and computes its co-occurrence matrices using the `compute cooccurrence matrices` function. The resulting tensor is returned as the preprocessed image.

**ii. Data Generation:** The approach uses the Image Data Generator class from Keras to generate batches of preprocessed images. The generator applies the `preprocess image` function to each image before feeding it into the network. The data generator splits the data into training and validation sets using a validation split of 0.25.

**iii. Model Architecture:** The approach defines a function called `build model` which constructs the neural network model. The model is built using the Sequential

API from Keras. It consists of several convolutional layers followed by max pooling layers for feature extraction. The flattened features are then passed through fully connected layers with ReLU activation. The final layer uses sigmoid activation for binary classification. A tensor (stack of the 6 matrices computed from the 3 spatial co-occurrence matrices and 3 cross-band co-occurrence matrices) is fed to the following CNN model architecture:

(6 convolutional layers followed by a single fully-connected layer)

- A convolutional layer with 32 filters of size  $3 \times 3$  followed by a ReLU layer
- A convolutional layer with 32 filters of size  $5 \times 5$  followed by a max pooling layer
- A convolutional layer with 64 filters of size  $3 \times 3$  followed by a ReLU layer
- A convolutional layer with 64 filters of size  $5 \times 5$  followed by a max pooling layer
- A convolutional layer with 128 filters of size  $3 \times 3$ , followed by a ReLU layer
- A convolutional layer with 128 filters of size  $5 \times 5$  followed by a max pooling layer
- A dense layer (256 nodes) followed by a sigmoid layer.

**iv. Model Training:** The model is compiled with the Adam optimizer, binary cross-entropy loss, and accuracy as the evaluation metric. The training is performed using the `fit` method, with the data generated by the `train_flow` and `valid_flow` generators. The training process is monitored using callbacks such as early stopping and model checkpointing. The number of epochs and steps per epoch are defined to control the training duration.

**v. Evaluation and Visualization:** The approach visualizes the training process by plotting the accuracy and loss curves for both the training and validation sets. These plots provide insights into the model's performance and help in identifying overfitting or underfitting. The trained model is also saved for future use.

### 4.2.2 Using FFT

The goal of this approach is to classify images as "Real" or "Fake GAN generated" using a deep learning model. To achieve this, a Fast Fourier Transform (FFT) is applied on each of the RGB channels of the input images. The transformed data is then shifted to the center using FFT shift and normalized. Finally, the original RGB channels are replaced with the transformed and normalized channels, and the images are fed into a deep neural network for classification.

#### 4.2.2.1 Experimental Setup

**i. Data Preprocessing:** The `preprocess_image` function, which accepts an image as input, is defined first in the approach. The image is divided into its RGB channels using this function, and each channel is then subjected to FFT. After that, the processed data is normalized by multiplying it by the FFT's highest absolute value. The preprocessed image is created by recombining the normalized transformed channels.

**ii. Data Collection:** The approach creates batches of preprocessed photos using the `ImageDataGenerator` class from Keras. Before sending any images into the network, the generator runs each one through the `preprocess_image` function. The data generator splits the data into training and validation sets using a validation split of 0.25.

**iii. Model Architecture:** The `build_model` function, which creates the neural network model, is defined in the approach. The Sequential API of Keras is used to construct the model. It starts with a number of convolutional layers and then moves on to max pooling layers to extract features. Then, using ReLU activation, the flattened features are passed through fully connected layers. Binary categorization is done using sigmoid activation in the top layer.

**iv. Model Training:** The Adam optimizer, binary cross-entropy loss, and accuracy are used in the model's construction. Using the data produced by the `train_flow` and `valid_flow` generators, the training is carried out using the `fit` approach. Callbacks, such as model checkpointing and early halting, are used to keep track of the training process. The number of epochs and the number of steps per epoch are set in order to regulate the training time.

**v. Evaluation and Visualization:** By showing the accuracy and loss curves for both the training and validation sets, the approach illustrates the training procedure. These plots offer information about the model's performance and assist in spotting over- or underfitting. Additionally saved for later use is the trained model.

### 4.3. Frequency Analysis

In this method we investigate the frequency domain of the GAN generated images to identify artifacts. To analyze the images in frequency domain, we use the discrete cosine

transform (DCT). The DCT is similar to Discrete Fourier Transform (DFT) where it sums cosine functions that oscillate at various frequencies to represent a finite sequence of data points. The DCT is used because it effectively compresses the energy in image signal and can be separated for effective implementations. Since, Style GAN generated images exhibit artifacts in frequency spectrum, the aim is to analyze if there is any common occurrence of pattern in Style GAN images by training the DCT transformed images using convolutional neural network.

#### 4.3.1 Experiment Setup

We used 10,000 GAN generated images and 10,000 real images from Celeb-A dataset for this experiment. We split the 10,000 images into 8,000 for training and 2,000 for validation. For training, the images are loaded and converted to gray scale. Then the images are converted using DCT and is resized to 256x256. The images are then normalized to the range [0,1]. For training the images, convolutional neural network using the keras API in python is initialized. The model contains two convolutional layers followed by max pool layer. The first convolutional layer has 32 filters and the second convolutional layer contains 64 filters. The output of second max pool layer is flattened and passed to two fully connected layers (dense) with 128 and 1 neurons. The final layer uses sigmoid activation function to output a binary classification. The model is trained with 20 epochs with an early stopping callback that checks the validation loss and stops training if it does not decrease over the course of five iterations. The model is then evaluated on a test set. The grey scale image is converted using DCT and is normalized to the range [0,1]. This image is then tested on the pre trained convolutional neural network and the predicted result is obtained.

### 4.4. Face feature Analysis

In this method we use the facial landmark positions to expose GAN synthesized images. The approach is based on the observation that the facial components configuration produced by GAN models differs from that of real faces since there aren't any global limitations. This approach is a super version of approach 2. This approach is more detailed and comprehensive that uses 32 facial points instead of 2 points. A SVM classifier is trained on the facial landmark locations to classify GAN synthesized faces from the real face.

#### 4.4.1 Experiment Setup

For our experiment, we utilized a dataset comprising of 10,000 images, including 10,000 GAN generated images and 10,000 real images obtained from Celeb-A dataset. We partitioned this dataset into two sets, 8,000 images for training and 2,000 images for validation. First, we iterate over each image and convert the image to grayscale. Then the face is detected using the landmark predictor from dlib. 51 facial landmark locations are extracted and is normalized to the  $[0,1] \times [0,1]$  region. After processing all the images, the result is stored into a file for the SVM classifier to train.

The feature vectors and labels of train and test set are split into 80:20 ratio using train\_test\_split function from the sklearn model selection. A grid search is used to perform exhaustive search over grid parameters to find the optimal hyperparameters for the SVM classifier. 5-fold cross validation object is used to ensure balance between the training and validation sets. The SVM classifier is trained with the feature vectors and corresponding labels using the method of the grid search object.

The model is then tested with profile images of Instagram public account. Using Instagram API, user profile information is fetched including user profile picture by providing Instagram user ID. The landmark facial key points are detected using the dlib library. The predicted landmark are combined and then normalized to  $[0,1] \times [0,1]$  region. The profile picture is determined GAN or not using the normalized feature vectors by the pre trained SVM model.

## 5. Results



**Fig. 3:** The green dots at eye specify the exact coordinate of the detected eyes in the image (Method 4.1)



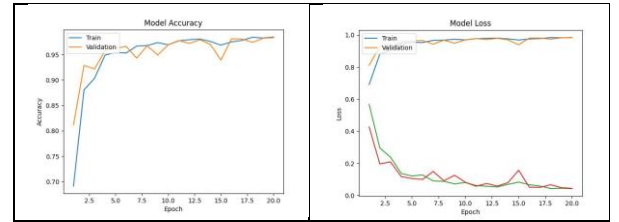
**Fig. 4:** 51 Facial landmark points detected for given image using Dlib library (Method 4.4)

<b>Best hyperparameters</b>	{'C': 10, 'gamma': 1}
<b>Accuracy</b>	0.9997

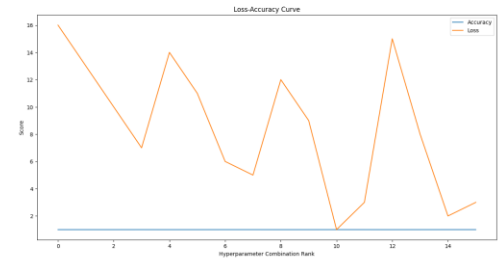
**Fig. 5:** Best hyper parameters and accuracy for SVM model. (Method 4.4)

Method	Validation Accuracy
4.1	N/A
4.2.1	95.74%
4.2.2	98.48%
4.3	99.35%
4.4	99.97%

**Table 1:** Validation accuracy for each approach

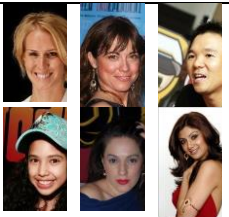



**Fig. 6:** Model Accuracy and Loss Plots for Method 4.2.2





**Fig. 7:** Model Loss Plots for SVM model (Method 4.4)

## 6. Discussion

	Predicted Real	Predicted GAN
Real		
GAN		

**Fig. 8:** Examples of correct and incorrect predictions by SVM Model (Method 4.4)

	Predicted Real	Predicted GAN
Real		
GAN		

**Fig. 9:** Examples of correct and incorrect predictions by CNN Model (Method 4.3)

	Predicted Real	Predicted GAN
Real		
GAN		

**Fig. 10:** Examples of correct and incorrect predictions by CNN Model (Method 4.4)

From Fig8, It can be seen that svm model predicted the all the correct labels. Where the CNN model for both Method 4.3 and Method 4.4 miss classified a real image into GAN image (fig 9 and fig 10)

Since Facial landmark approach using SVM model (Method 4.4) has more number of facial points for a given image compared to Method 4.1 which has just 2 points. We implemented Method 4.4 over method 4.1.

## 7. Conclusion

To summarize, four distinct methods were utilized to identify images generated by StyleGAN. The SVM classifier, which relied on facial feature landmark positioning, yielded the least accurate results. On the other hand, frequency analysis approaches provided the most accurate results, with crossband cooccurrence analysis methods coming in a close second

## 8. Future Directions

Generalization to other GAN architectures: The models developed for detecting StyleGAN generated images could be adapted to other GAN architectures. This would involve training the models on images generated by different GAN architectures, which could improve the models' ability to distinguish between real and generated images.

To improve the crossband cooccurrence analysis approach, researchers can explore the use of higher-order statistical analysis techniques. This would allow for a more detailed analysis of the correlations between different frequency bands, which could improve the accuracy of the detection.

## References

- [1] "Detecting GAN-Generated Images Using Color Coherence Vectors" by Abdullah Hamdi et al.
- [2] "Detecting GAN-Generated Images Using Convolutional Neural Networks" by Dang et al.
- [3] "A Novel Neural Model based Framework for Detection of GAN Generated Fake Images" by S. Agarwal, N. Girdhar and H. Raghav et al.
- [4] "West, J. and Bergstrom, C. Which face is real? <http://www.whichfaceisreal.com>, 2019."
- [5] "Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019."