PAPER NAME
C004_C022_C023_C024_REPORT.docx

AUTHOR
Hetvi Gandhi

WORD COUNT
3489 Words

CHARACTER COUNT
19513 Characters

PAGE COUNT
13 Pages

FILE SIZE
3.1MB

SUBMISSION DATE
Apr 2, 2024 5:54 PM GMT+5:30

REPORT DATE
Apr 2, 2024 5:54 PM GMT+5:30

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 8% Publications database
- Crossref database
- Crossref Posted Content database

Excluded from Similarity Report

- Internet database
- Submitted Works database
- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 8 words)

Synthesising Image Data

Project Report submitted in the partial fulfilment
Of

Bachelor of Technology

In

Computer Science

By

Rounak Bachwani (C004),
Riddhi Dumre (C022),
Nilay Gaitonde (C023),
Hetvi Gandhi (C024),
Vishesh Giyanani (C027)

Under the supervision of

Abhay Kolhe

(Professor, MPSTME)

Archana Nanade

(AsstProfessor, MPSTME)

SVKM's NMIMS University

Table of Contents

Chapter 1: Introduction

1.1 Background of the project topic

In recent years, computer vision is one of the many fields that have seen striking advancements due to the rapid growth in deep learning and machine learning algorithms. One area that has gotten significant attention is the generation of synthesized data due to its important and useful applications like data augmentation for training models and privacy-preserving data generation. This project aims to leverage the power of Generative Adversarial Networks (GANs), object detection, and image classification to create labeled synthesized face images.

By combining GANs with object detection and image classification, we aim to generate synthesized face images with precise labels, enabling the development of an accurate facial recognition system. This report presents the challenges and outcomes achieved by our model.

Chapter 2 : Literature survey

2.1 Introduction to the topic

In our projects we use 3 different types of models, GANs to create synthesized face images, an object detection model to detect a person in the image and a classification model to distinguish between real and fake faces. Similarly the related work section will be divided into 3 sections.

1. Object detection

   Object detection is a vital computer vision task, with deep learning models taking center stage. Traditional methods include two-stage detectors like R-CNN, Fast R-CNN, MatCNN and Faster R-CNN, which employ selective region proposals, and single-stage detectors that simultaneously detect objects across all spatial regions. Among the single-stage detectors, the YOLO (You Only Look Once) series has gained traction due to its balanced combination of speed and accuracy, leveraging convolutional neural networks (CNNs) for feature extraction.The YOLO architecture has evolved from version 1 to 4, introducing improvements such as batch normalization, anchor boxes, and multi-scale training [9]. However, challenges persist, including multi-scale training, class imbalance, detection of small objects, and the need for large datasets [10]. Tausif Diwan et al. [10] provide a comprehensive review of single-stage detectors, focusing on the YOLO series, and suggest future research directions, such as exploring advancements in CNNs and optimization techniques.

2. Generative Adversarial Network
   a. From "A Hybrid Model for Identity Obfuscation by Face Replacement", we utilized improved evaluation metrics like LPIPS and SSIM, beyond just FID, for more comprehensive assessment of perceptual and structural similarity.

   b. Inspired by "Learning to Anonymize Faces for Privacy Preserving Action Detection", we extended face anonymization to also cover other identifying features like body shapes and clothing for comprehensive privacy protection.

   c. While "Face Image Anonymization via Multidimensional Data K-Anonymizer" demonstrated effective neural network-based face anonymization, we tested on more

diverse datasets with complex scenarios and multiple individuals to thoroughly evaluate generalizability.

3. Image Classification

   a. The paper explores facial manipulation techniques, with a focus on DeepFakes, and the methods used to detect fake content. It covers four main types of facial manipulation, discusses the impact of Generative Adversarial Networks (GANs) on creating realistic fake content, and emphasizes the importance of public databases and benchmarks for evaluating detection methods.

   b. The study proposed a Recurrent Convolutional Network (RCN) model for detecting deepfake images by combining Convolutional Neural Networks and Recurrent Neural Networks, evaluated on the FaceForensics++ dataset containing 1,000 videos, and leveraged intraframe discrepancies and temporal anomalies between frames.

   c. The authors conducted an in-depth exploratory study to assess the generalizability of various fake face detection methods, encompassing both state-of-the-art deep learning techniques and conventional texture descriptor-based approaches. To facilitate this generalizability analysis, they introduced a novel database consisting of fake faces generated using diverse methodologies.

2.2 Problem Statement:

Despite the widespread applications of facial recognition technology, finding a diverse and accurately labeled facial image dataset remains a challenge. We come across numerous challenges alongside it, like the time-consuming and costly labor of collecting and annotating traditional data and privacy concerns arising from the collection of real-world facial images.

In our project, we aim to address these challenges by generating labeled synthesized face images using Generative Adversarial Networks (GANs), object detection, and image classification techniques. By generating diverse facial images with accurate labels, we overcome the challenges of traditional data collection methods.

Chapter 3 : Proposed System

In this project we propose a system that is able to generate synthesized data using GANs. We also are using an image classification model based on the DenseNet model to classify between fake and real images to not only be able to provide real and fake labels but also as a litmus test to prove the accuracy of a GAN. As you can see in Fig. 1 we first take the user's camera feed as input to check if a person is detected in the image to reduce malicious attacks by bots. Once the presence of a person is verified we then use the GAN to create a set of 4 images (this can be changed as and when needed). The GAN's output is then fed to the classification model for labeling.

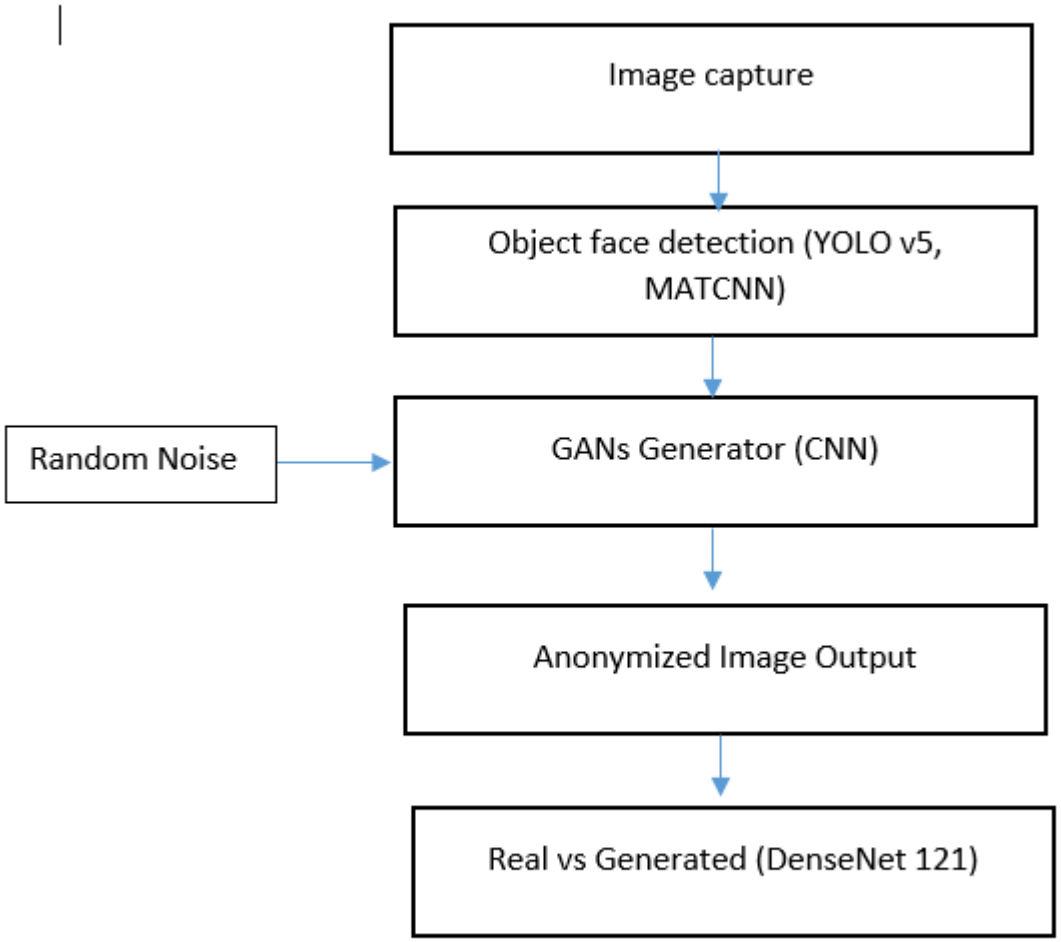


Fig 1: YOLO-GANs-DenseNet Model architecture

Chapter 4 : Methodology and Implementation

4.1 Object Detection

In our project we use the pre-built YOLO model to understand and detect the person from their camera feed. We use a pre-built method as object detection is a very vast and complex field of Deep Learning. Creating these models is extremely time and resource consuming and we found in our research even with thorough understanding and implementation we were able to achieve very low levels of accuracy. To avoid mishaps we use the state-of-the-art YOLO model. The YOLO model has an architecture similar to GoogleNet in that they both aim to create a single unified model.

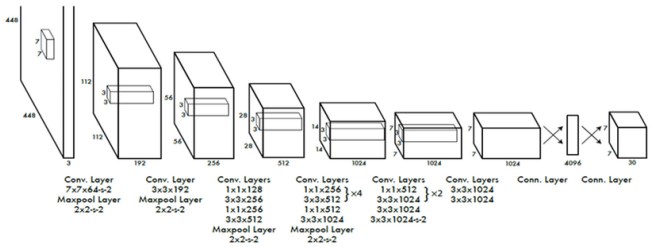


Fig 2 - YOLO model architecture

We can see in the above image the model has a single-stage object detection architecture that divides the input image into an n x n grid of cells, where each cell predicts a fixed number of bounding boxes and their associated class probabilities, along with an "objectness" score representing the likelihood of an object being present. The convolutional base extracts feature maps from the image, which are then processed by the detection layers to generate these predictions. After the network outputs the bounding boxes and confidence scores, a non-maximum suppression algorithm filters out overlapping boxes, keeping only the most confident predictions.

With the YOLO model we have also used the MTCNN model for better accuracy. The MTCNN progressively refines its predictions by first proposing candidate face regions, then refining these proposals through bounding box regression, and finally performing facial landmark localization to accurately align the detected faces. Each stage of the network consists of convolutional layers followed by non-linear activation functions and pooling layers, allowing it to effectively capture hierarchical features at multiple scales.

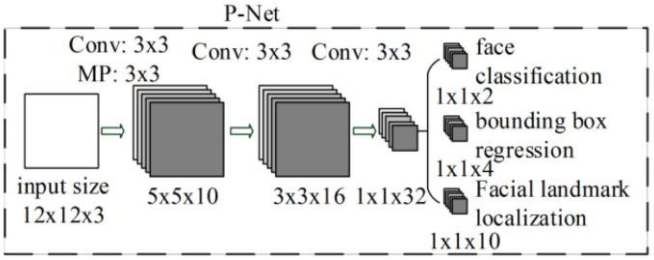


Fig 3 - MTCNN architecture

4.2 GAN Image Generator

The generator model aims to generate or synthesize new data samples, such as images, based on the learned distribution from the training data. The model mainly uses convolution layers, batch normalization layers to normalize the output and leaky ReLU as the activation function. The model starts with a lower-dimensional input and gradually upscales or increases the spatial dimensions through transpose convolutions (Conv2DTranspose) and reshape operations.

The discriminator model, commonly used in GANs to distinguish between real and fake data samples. The discriminator model takes an input, such as an image, and aims to classify it as either real or fake. The discriminator architecture follows a more traditional convolutional neural network structure, with convolutional layers, batch normalization layers , and leaky ReLU activation. The model performs gradual downsampling or reduction of spatial dimensions, till it reaches a final image that is given to a dense layer which represents the probability of a given image being fake or real.

Fig 4 - GANs Architecture

4.3 Image Classification

The object classification model is again a pre-built DenseNet model that classifies images as real or fake. DenseNet121 is a convolutional neural network architecture designed for image classification tasks. It consists of multiple dense blocks where each layer is connected to every other layer in a feed-forward fashion. These dense blocks promote feature reuse and facilitate gradient flow throughout the network, enhancing parameter efficiency and enabling deeper architectures. Each dense block is composed of convolutional layers followed by batch normalization and ReLU activation, with transition layers in between to reduce spatial dimensions. DenseNet121 specifically has 121 layers, with a global average pooling layer and a fully connected layer at the end for classification. This architecture's densely connected structure allows it to effectively capture intricate patterns in images while maintaining relatively fewer parameters compared to traditional convolutional networks.

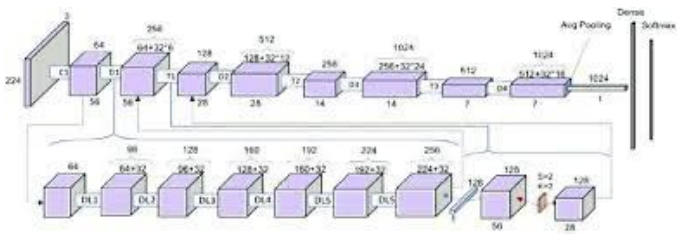


Fig 5 - DenseNet 121 architecture

Chapter 5

Code and Screenshots

The final code is available on github at https://github.com/rounaccc/MLOA-NNDL-Project

5.1 Process

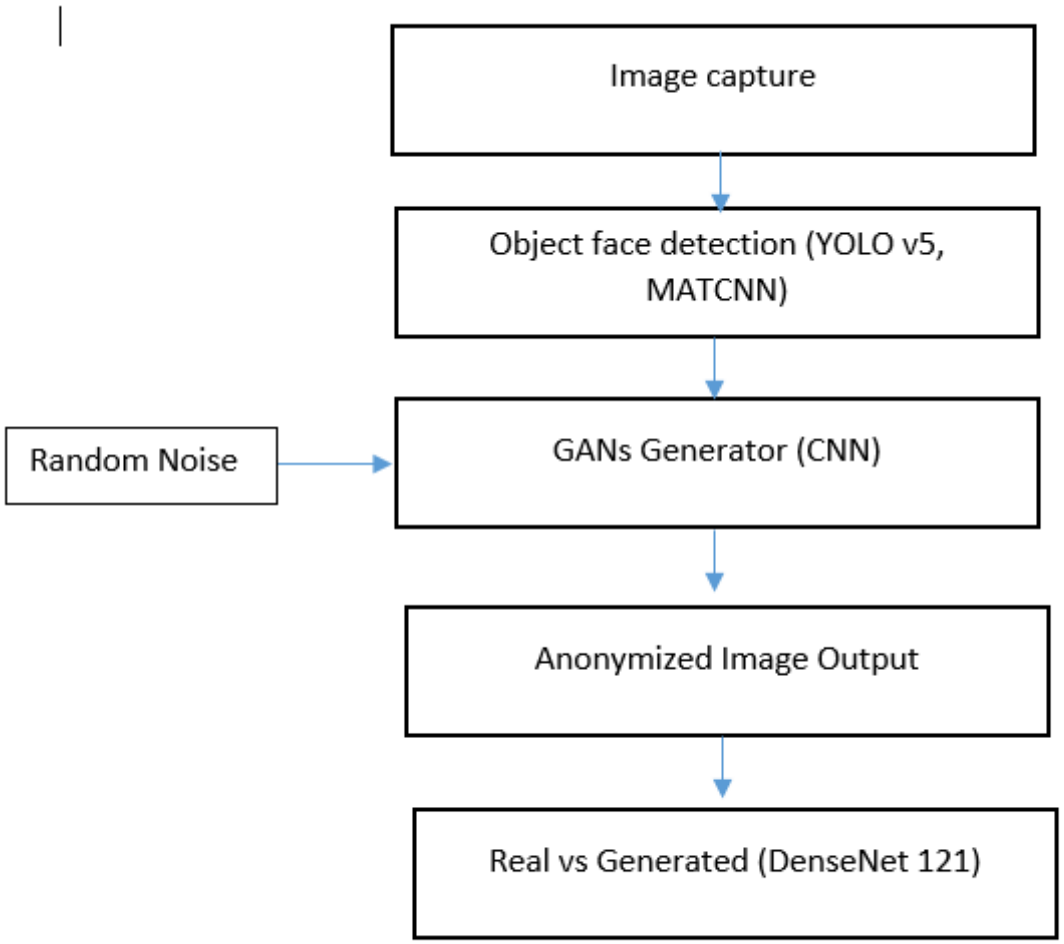


Fig 1: YOLO-GANs-DenseNet Model architecture

Figure 1 provides an overview of the flow of the architecture proposed in this study. Commencing with the intake of a color image, we leverage the robust capabilities of YOLO v5 to detect pertinent objects within the scene. Upon identification of a human subject, we employ a generator network, complemented by a random noise input of dimensions (128, 128) and a latent dimension of 100, to produce an anonymized rendition of the image.

This anonymized image undergoes rigorous scrutiny through our third model, DenseNet, which performs a discriminating task of classifying images as either real or generated. This

critical step ensures the integrity of the anonymization process, facilitating the preservation of privacy while maintaining the fidelity of the processed image.

5.2 Object Detection

For object detection we compare the accuracy of two object detection models: MTCNN (Multi-task Cascaded Convolutional Networks) and YOLOv5 (You Only Look Once version 5). The comparison is done on a single image.The function compare_models is defined to perform this comparison. It takes two arguments: filename and pixels. filename is the name of the image file, and pixels is the image data in the form of a NumPy array. The function starts by using the detect_faces function of the detector object to detect faces in the image. The detector object is not defined in the provided code, but it's likely an instance of the MTCNN model. The detect_faces function returns a list of detected faces, each represented as a dictionary with details about the face. The accuracy of the MTCNN model is then calculated as the average confidence of the detected faces. Next, the function loads the YOLOv5 model from the Ultralytics GitHub repository using the torch.hub.load function. The model is used to detect objects in the image.

The model function returns a results object that contains the detection results. The function then filters the results to only keep the detections of persons. The accuracy of the YOLOv5 model is calculated as the average confidence of the detected persons. The function then compares the accuracies of the two models. If the MTCNN model has a higher accuracy, it prints a message stating that MTCNN has a higher accuracy for face detection, and sets model_name to 'MTCNN'. Otherwise, it prints a message stating that YOLOv5 has a higher accuracy for person detection, shows the detection results using the results.show function, and sets model_name to 'YOLOv5'. Finally, the function returns the accuracies of the two models and the name of the model with the higher accuracy.

The script also contains a main section that is executed when the script is run as a standalone program. In this section, it reads an image file using the pyplot.imread function, and calls the compare_models function to compare the accuracies of the two models on this image. The accuracies are then stored in the mtcnn_accuracy and yolov5_accuracy variables.

5.3 GAN

The Generator function in the code defines the architecture of the generator part of the GAN. It takes as input a parameter latent_dim, which refers to the size of the latent space. The latent space is a compressed representation of the data, and the generator will use this to generate new data instances. The generator is a Sequential model from the Keras API. It starts with a Dense layer that takes the latent space as input and outputs a flattened version of the image. This is then reshaped into the shape of the image (128x128x3). The model then goes through a series of convolutional layers (Conv2D), batch normalization layers (BatchNormalization), and leaky ReLU activation layers (LeakyReLU). The convolutional layers are used to extract features from the images, while the batch normalization layers normalize the activations of the previous layer, which can speed up the learning process and reduce the chance of getting stuck in local minima during training. The leaky ReLU activation function is used to add non-linearity to the model and prevent the problem of dying ReLUs (where ReLU neurons essentially become inactive and only output 0).

The model also includes Conv2DTranspose layers, which perform a transposed convolution operation, also known as a deconvolution. This is essentially the reverse of a convolution operation and is used to increase the spatial dimensions of the output — this is part of the upsampling process to generate a new image. Finally, the model returns an image with the same size as the input image but with the pixel values squashed between -1 and 1 due to the tanh activation function. This is the generated image that will be passed to the discriminator for evaluation.

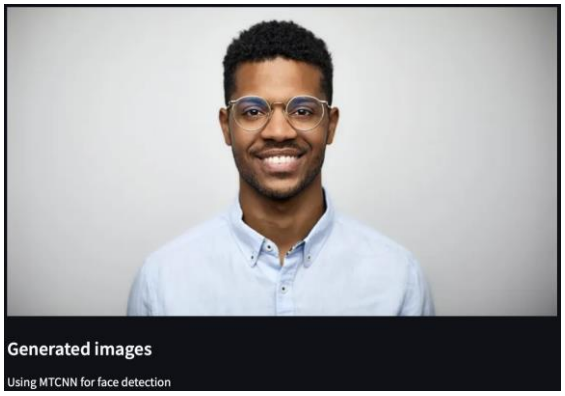
5.4 Image Classification

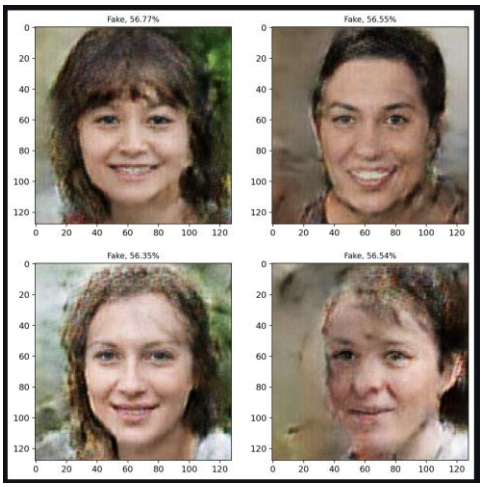
The preprocess_image(image_path) function is used to preprocess the input image. It takes an image path as input, loads the image in grayscale and resizes it to 224x224 pixels. The image is then converted to an array and normalized by dividing each pixel value by 255.0 (this is done to bring the pixel values between 0 and 1). The image array is then expanded by adding an extra dimension at the beginning, which is a common requirement for models in Keras that expect a batch of images as input, not a single image. The preprocessed image array is returned as output. The make_prediction(model, image_path) function is used to make a prediction on the preprocessed image. It takes a pre-trained model and an image path as input. The image is preprocessed using the preprocess_image(image_path) function and then passed to the model for prediction. The prediction is printed to the console. If the prediction value is greater than 0.7, the function returns "Real" along with the prediction value; otherwise, it returns "Fake" along with the prediction value.

While execution the script first loads the pre-trained model from the specified path. Then it sets the image path to be used for prediction. It checks if the image path is valid; if not, it prints an error message and returns. If the image path is valid, it calls the make_prediction(model, image_path) function to make a prediction on the image. The prediction result is then printed to the console.

5.5 Streamlit implementation

The Streamlit application allows users to upload an image for processing and analysis using machine learning models. The application loads pre-trained models for image generation, object detection, and real vs. fake image detection. The user interface prompts the user to either upload an image or try a demo. If the user chooses to upload an image, the image is captured and processed. If a human face is detected in the image, the application generates a set of images and displays them, each labeled with a prediction of whether it's real or fake. If no human face is detected, the application notifies the user. If the user chooses to try a demo, the application loads a demo image, displays it, and performs the same image generation and real vs. fake prediction as with the uploaded image.





1. The system starts by taking a color image as the initial input.
2. An object detection model called YOLO v5 (You Only Look Once) is employed to scan the input image and identify any objects present. Specifically, the model checks if there are any human subjects in the image.
3. If a human is detected:
    - The image gets passed to a generator network, which is part of a generative adversarial network (GAN) architecture.
    - Along with the original image, the generator takes a random noise vector with dimensions of (128, 128) and a latent dimension of 100.
    - Using these inputs, the generator creates a new, synthetic version of the image where the detected human's identity is concealed or altered in some way, effectively anonymizing them.
4. The anonymized image from the previous step is then evaluated by a separate model called DenseNet:
    - DenseNet acts as a discriminator, analyzing the image.
    - Its task is to classify whether the input image is an original, unaltered image (real) or if it's synthetically generated.
    - This verification step assesses the quality and effectiveness of the anonymization process, ensuring that privacy is protected while still preserving the overall usability and fidelity of the image content.

Chapter 6 : Conclusion and Future Work

6.1 Conclusion

The proposed system leverages the power of Generative Adversarial Networks (GANs) to generate synthesized image data, addressing the common challenge of limited data availability. The generated images are then classified as real or fake using a pre-trained DenseNet model, providing a mechanism to evaluate the performance of the GAN while also generating labeled synthetic data.

The system incorporates an initial person detection step using the YOLO model to mitigate potential malicious attacks from bots and ensure that the input data is relevant. The GAN model, consisting of a generator and a discriminator, generates a set of four synthetic images, which are then fed into the classification model for labeling as real or fake.

This approach not only facilitates the generation of synthetic data but also serves as a litmus test to assess the accuracy and capability of the GAN model in producing realistic samples. The combination of these state-of-the-art models aims to provide a comprehensive solution for data augmentation, privacy-preserving data generation, and model testing across various domains.

6.2 Future Work

- Expand the system's capabilities to generate and classify other data types, such as text, audio, or video, by adapting or developing specialized GAN architectures and classification models.
- Improve the GAN's performance through advanced architectures, training strategies, or loss functions, enhancing the quality and diversity of the generated synthetic data.

- Explore conditional GANs to enable more controlled and targeted data generation based on specific conditions or attributes.
- Investigate the integration of semi-supervised or unsupervised learning techniques to leverage the generated synthetic data for improving the performance of object detection and classification models.
- Develop comprehensive evaluation metrics tailored to specific applications to assess the quality and usefulness of the generated data more effectively.
- Address potential ethical concerns, such as privacy, bias, and misuse, by developing guidelines and best practices for the responsible and ethical use of synthetic data generation techniques.
- Through continuous research and development in these areas, the proposed system can be enhanced and expanded, unlocking new possibilities in data generation, augmentation, and model evaluation across various fields that rely on high-quality and diverse data.

Chapter 7 : References

[1] T. Nakamura, Y. Sakuma and H. Nishi, "Face Image Anonymization as an Application of Multidimensional Data K-Anonymizer," 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW), Nagasaki, Japan, 2019, pp. 155-161, doi: 10.1109/CANDARW.2019.00035. keywords: {K-Anonymization;Face Image;High-Dimension;StyleGAN;Neural Network},

[2] Sun, Qianru et al. “A Hybrid Model for Identity Obfuscation by Face Replacement.” ArXivabs/1804.04779 (2018): n. pag.

[3] Ren, Zhongzheng & Lee, Yong & Ryoo, Michael. (2018). Learning to Anonymize Faces for Privacy Preserving Action Detection.

[4] Khodabakhsh, Ali & Ramachandra, Raghavendra & Raja, Kiran & Wasnik, Pankaj & Busch, Christoph. (2018). Fake Face Detection Methods: Can They Be Generalized?. 10.23919/BIOSIG.2018.8553251.

[5] Hasin Shahed Shad, Md. Mashfiq Rizvee, Nishat Tasnim Roza, S. M. Ahsanul Hoq, Mohammad Monirujjaman Khan, Arjun Singh, Atef Zaguia, Sami Bourouis, "[Retracted] Comparative Analysis of Deepfake Image Detection Method Using Convolutional Neural Network", Computational Intelligence and Neuroscience, vol. 2021, Article ID 3111676, 18 pages, 2021. https://doi.org/10.1155/2021/3111676

[6] Tolosana, Ruben, et al. "Deepfakes and beyond: A survey of face manipulation and fake detection." Information Fusion 64 (2020): 131-148.

[7] Diwan, T., Anirudh, G. & Tembhurne, J.V. Object detection using YOLO: challenges, architectural successors, datasets and applications. Multimed Tools Appl 82, 9243–9275 (2023). https://doi.org/10.1007/s11042-022-13644-y