Deepfake Detection Using GAN Discriminators: Implementation and Result Analysis

Gaurav Singh Bisht[1], Pratik Gavit[2], Arnav Godhamgaonkar[3], Harshil Poshiya[4], Prof. J. S. Pawar[5]
[1,2,3,4] *Department of Information Technology, Sinhgad College of Engineering, Pune 411041, India*
[5]*Professor, Department of Information Technology, Sinhgad College of Engineering, Pune 411041, India*

***Abstract*— Differentiating between actual and deepfakes becomes increasingly challenging in digital forensics as synthetic picture production advances. This paper offers a practical approach to detect images generated by a Vanilla Generative Adversarial Network (GAN) by use of the discriminator of another vanilla generator. Since the GAN discriminator is adversarially trained to uncover minute deviations from true data distributions, it is naturally suited for deepfake detection unlike traditional classifiers. Following Vanilla GAN training, we ran a fine-tuning step in which the discriminator acted as a stand-alone classifier. In controlled settings the model achieved a final detection accuracy of 100%. This work significantly clarifies the possibilities of adversarial components as forensic tools by offering information on the technique, architecture, training phases, and performance results. Future directions demand evaluating cross-GAN generalization and enhancement of model robustness for pragmatic environments.**

Keywords*— *Deepfake Detection, Generative Adversarial Networks (GANs), GAN Discriminator, Synthetic Image Classification, Adversarial Training, Vanilla GANs, Discriminator Fine-Tuning.

I. INTRODUCTION

Generative Adversarial Networks (GANs) have become among the most important developments in artificial intelligence recently. Comprising a generator and a discriminator taught in opposition, these models have shown amazing success creating hyper-realistic images, audio, and video material. Although their contributions to fields including creative media, medical imaging, and simulation are noteworthy, their use has also raised questions about abuse—most famously in the form of deepfakes. Often without permission, deepfakes—synthetic images or videos that convincingly copy real people—may compromise information authenticity, privacy, and trust.

Finding such synthetic media gets more difficult as the realism of GAN-generated content keeps improving. Currently used deepfake detection methods mostly rely on CNN-based classifiers educated on labeled datasets of real and synthetic images. Although these methods produce good results on known data, they sometimes cannot generalize over various GAN architectures or datasets not observed during training. This restriction drives a great demand for more flexible and architecture-agnostic detection systems able to detect deepfakes in dynamic and various environments.

We present in this work a fresh but simple method: using the discriminator from a trained Vanilla GAN as a detection engine for images produced by GANs. We speculate that the discriminator has latent ability for deepfake detection since it is naturally optimized to differentiate real from fake samples during adversarial training. First we separate and fine-tune the discriminator for binary classification after training a Vanilla GAN using conventional adversarial techniques. Our results show that this basic framework can attain high accuracy in controlled experiments, implying that GAN discriminators—often neglected after training—can be lightweight, efficient detectors of synthetic content [1], [2].

II. LITERATURE REVIEW

Generative Adversarial Networks' (GANs') development has revolutionized synthetic media generation's field. Deepfakes are becoming more and more similar to real content, thus parallel development in detection techniques has started to counteract their use. Conventions for deepfake detection mostly rely on frequency-based analysis or Convolutional Neural Networks (CNNs). These

approaches, however, sometimes fail to generalize across various GAN architectures or image domains, which results in brittle detection frameworks when used in practical settings.[3]

Emphasizing that deep learning models trained on frequency artifacts and color anomalies typically outperform passive forensics but still lack transferability across domains, Remya Revi et al.[4] surveyed GAN-based deepfake detection methods. Abdullah et al.[5] underlined similar limits in detection generalization and suggested architectural improvements depending on content-agnostic feature extraction. They also demanded models who could change with the times to fit changing deepfake generating techniques.

Yang et al.,[6] who first proposed the idea of a dynamic discriminator, go over a particularly pertinent method here. Their work suggested that the capacity of a discriminator should change during training to correspond with the different generator output difficulty. Without adding computational complexity, this adaptive capacity model—DynamicD—showered better performance in both synthesis quality and downstream classification. Their work emphasizes training dynamics, but it also supports the notion that discriminators have deep representational insights fit for use in detection tasks.

Moreover, basic research by Goodfellow et al.[7] established the theoretical framework for adversarial networks, in which the generator is guided toward realism by means of imperfection identification in synthetic data, hence the discriminator is central. Although most studies concentrate on generator enhancements, more recently conducted studies have turned their focus to using discriminator knowledge after training. One such research proposes to use pre-trained discriminators as forensic tools, especially when refined using supervised learning to differentiate between real and fake images.

With PCA and SVMs, Hanady S. A. Kareem et al.[8] presented a machine learning-based method for deepfake face classification attaining over 96% accuracy. Although their pipelines are different, the use of reduced feature representations is similar to how GAN discriminators isolate important distinguishing features by adversarial learning. Focusing on small discrepancies brought about during image synthesis, further works by Li et al. and McCloskey et al.[9] investigated color-space transformations and high-pass filtering. These methods confirm the relevance of fine-grained artifact detection—a natural ability of adversarially trained discriminators—even if they are not directly related to GAN discriminators.

All things considered, the literature shows increasing awareness of GAN discriminators as rich sources of forensic information. Still underexplored, though, their reusing outside adversarial training. This work fills in that void by suggesting a structured pipeline isolating and optimizing a Vanilla GAN discriminator to operate as a stand-alone deepfake detector. We want to use the embedded feature space of the discriminator to create a lightweight, architecture-agnostic detection framework by merging adversarial training with focused fine-tuning.

III. PROBLEM STATEMENT

Generative Adversarial Networks (GANs) have significantly advanced the field of synthetic media generation. The rapid development of generative models has led to the widespread creation of highly realistic synthetic media, commonly referred to as deepfakes. These synthetic images and videos can convincingly replicate human faces, expressions, and environments, making them increasingly difficult to distinguish from real content using traditional inspection or forensic techniques. While this innovation has enabled valuable applications in art, medicine, and education, it also poses significant risks, including the spread of misinformation, identity fraud, and erosion of public trust in digital content.

To counter these threats, numerous deepfake detection methods have been developed, primarily relying on frequency spectrum anomalies, artifact-based forensic cues, or supervised learning techniques employing convolutional neural networks (CNNs). However, these models often face two major limitations. First, they tend to be domain-specific, tailored to detect fakes from particular datasets or GAN architectures (e.g., StyleGAN, CycleGAN), and struggle to generalize to images produced by other techniques. Second, training these models from scratch or maintaining large ensembles for robust detection incurs substantial computational and data costs, making them less practical for lightweight or real-time deployment.

Each GAN inherently includes a discriminator network trained to differentiate between real images and those generated by its corresponding generator. This discriminator is adversarially optimized by continuously analyzing real and synthetic data to identify subtle differences between the two distributions. Despite its central role in GAN training, the discriminator is seldom repurposed for real-world applications, particularly in detection tasks. This underutilization represents a missed opportunity, as the discriminator possesses a learned representation of synthetic features that could be leveraged in downstream classification problems [10].

This study explores the potential of employing a trained discriminator as an independent deepfake detection engine to address the underutilization of GAN discriminators in post-training scenarios. Specifically, we examine the effectiveness of fine-tuning the discriminator from a trained Vanilla GAN for binary classification tasks distinguishing real images from GAN-generated ones. Our objective is to determine whether the latent knowledge embedded in the discriminator—originally shaped through adversarial training—can serve as a rapid, accurate, and broadly applicable tool for identifying synthetic visual data [11].

The primary challenge lies in the absence of lightweight, reusable detection models that do not require architectural overhauls or extensive retraining. This work proposes a novel yet straightforward pipeline to bridge this gap by validating the GAN discriminator as a viable solution for detecting deepfake images, potentially reducing computational overhead and enhancing the adaptability of detection systems across various generative environments.

IV. METHODOLOGY

The design, implementation, and workflow of our suggested method—repurposing a Vanilla GAN discriminator for the detection of GAN-generated (fake) images—are described in this section. Architectural design, dataset curation, adversarial training, discriminator fine-tuning, and performance evaluation are the five main phases that make up the methodology.

4.1 GAN-Based Detection Framework Overview

A Vanilla Generative Adversarial Network (GAN), which is the basis of this work, is made up of two neural networks—a discriminator and a generator—that have been trained in opposition to one another. While the discriminator is trained to discern between authentic and fraudulent inputs, the generator learns to create progressively more realistic images from random noise during adversarial training. We decouple the discriminator from the GAN framework after the adversarial training is finished, and then retrain it under supervision to identify real or fake images.

Compared to conventional deepfake detection pipelines, which train classifiers from scratch on labeled datasets, this method is essentially different. Here, the discriminator has a solid representational foundation for classification because it has already been exposed to both kinds of data during adversarial training [10], [11].

4.2 Architectural Details

Generator Architecture

Early in the generator network, a dense layer expands the noise vector into a high-dimensional tensor, which is then transformed into an initial feature map. Four Conv2DTranspose (deconvolution) layers—each with LeakyReLU activations and Batch Normalization—follow next. To create 128×128 grayscale images, the last layer scales pixel values between -1 and 1 using a Tanh activation function. The generator boasts roughly 2.78 million trainable parameters overall.

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 64, 64, 64)	1088
leaky_re_lu_16 (LeakyReLU)	(None, 64, 64, 64)	0
dropout_8 (Dropout)	(None, 64, 64, 64)	0
conv2d_9 (Conv2D)	(None, 32, 32, 128)	131200
leaky_re_lu_17 (LeakyReLU)	(None, 32, 32, 128)	0
dropout_9 (Dropout)	(None, 32, 32, 128)	0
conv2d_10 (Conv2D)	(None, 16, 16, 256)	524544
leaky_re_lu_18 (LeakyReLU)	(None, 16, 16, 256)	0
dropout_10 (Dropout)	(None, 16, 16, 256)	0
conv2d_11 (Conv2D)	(None, 8, 8, 512)	2097664
leaky_re_lu_19 (LeakyReLU)	(None, 8, 8, 512)	0
dropout_11 (Dropout)	(None, 8, 8, 512)	0
flatten_2 (Flatten)	(None, 32768)	0
dense_4 (Dense)	(None, 1)	32769

Total params: 2787265 (10.63 MB)
Trainable params: 2787265 (10.63 MB)
Non-trainable params: 0 (0.00 Byte)

Fig 1. Generator architecture showing the transformation of a random noise vector into a 128×128 grayscale image using transposed convolutions and Tanh activation.

Discriminator Architecture

The discriminator might receive 128 x 128 x 1 grayscale image. LeakyReLU activations and Batch Normalization come after numerous Conv2D layers with progressively broader filter widths (64–512). The network finishes at a Dense layer generating a probability score reflecting the correctness of the input using a sigmoid activation function. Dropout layers help lower overfit. Total there are about 2.34 million parameters.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 16384)             1654784

reshape_2 (Reshape)          (None, 8, 8, 256)         0

leaky_re_lu_20 (LeakyReLU)   (None, 8, 8, 256)         0

batch_normalization_8 (Bat   (None, 8, 8, 256)         1024
chNormalization)

conv2d_transpose_8 (Conv2D   (None, 16, 16, 128)       524416
Transpose)

leaky_re_lu_21 (LeakyReLU)   (None, 16, 16, 128)       0

batch_normalization_9 (Bat   (None, 16, 16, 128)       512
chNormalization)

conv2d_transpose_9 (Conv2D   (None, 32, 32, 64)        131136
Transpose)

leaky_re_lu_22 (LeakyReLU)   (None, 32, 32, 64)        0

batch_normalization_10 (Ba   (None, 32, 32, 64)        256
tchNormalization)

conv2d_transpose_10 (Conv2   (None, 64, 64, 32)        32800
DTranspose)

leaky_re_lu_23 (LeakyReLU)   (None, 64, 64, 32)        0

batch_normalization_11 (Ba   (None, 64, 64, 32)        128
tchNormalization)

conv2d_transpose_11 (Conv2   (None, 128, 128, 1)       513
DTranspose)

=================================================================
Total params: 2345569 (8.95 MB)
Trainable params: 2344609 (8.94 MB)
Non-trainable params: 960 (3.75 KB)
_________________________________________________________________
```

Figure 2. Architecture of the discriminator network used in the Vanilla GAN, featuring stacked convolutional layers with batch normalization and LeakyReLU activations.

This architecture strikes a balance between complexity and interpretability, making it ideal for proof-of-concept detection pipelines.

4.3 Dataset Preparation and Preprocessing

To ensure consistent training and evaluation, we created a balanced dataset comprising generated as well as actual images. Real images were reduced from publicly available facial picture databases to 128x 128 pixels. We produced synthetic images using our Vanilla GAN's trained generator component. Once the model reached visual convergence—which came about after about 200 epochs—these synthetic images were kept and assigned suitable labels.

Among the chores related to preparation were:

1. All images were grayscale [11] to match the expected 128x128x1 input format.

2. Pixel values were min-max adjusted to the range [-1, 1] [12] so allowing constant training.

3. The dataset was randomly shuffled and split, with 80% set for training and 20% for testing, so guaranteeing a fair mix of actual and synthetic images [13].

This ongoing preprocessing ensured that the architectural needs of the network matched the phases of training and testing, so promoting effective learning and evaluation.

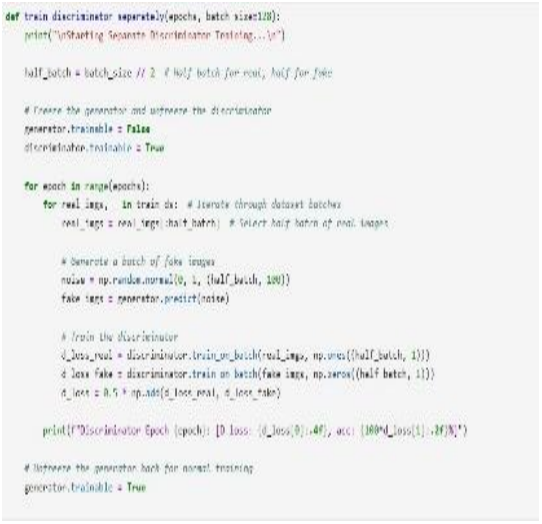
4.4 Two-Phase Training Process

Phase 1 – Adversarial Training (GAN Training):

Binary Crossentropy loss and the Adam optimizer (learning rate = 0.0002, $\beta_1 = 0.5$) drove us to train the GAN for 200 epochs. The generator wanted to limit the discriminator's capacity to identify fakes while the discriminator sought to maximize it in this phase, therefore engaging in a zero-sum game [14]. Typical with GANs, the training graphs displayed varying loss values together with slow increases in image quality and discriminator accuracy [15].

```
def build_generator():
    noise_shape = (100,)  # Latent space dimension
    model = Sequential()

    # Fully connected layer to reshape into a small spatial feature map
    model.add(Dense(8 * 8 * 256, input_shape=noise_shape))
    model.add(Reshape((8, 8, 256)))  # Reshape to (8,8,256)
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))

    # Upsampling layers
    model.add(Conv2DTranspose(128, kernel_size=4, strides=2, padding="same"))  # (16,16,128)
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))

    model.add(Conv2DTranspose(64, kernel_size=4, strides=2, padding="same"))  # (32,32,64)
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))

    model.add(Conv2DTranspose(32, kernel_size=4, strides=2, padding="same"))  # (64,64,32)
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))

    # Final layer to output 128x128 grayscale image
    model.add(Conv2DTranspose(1, kernel_size=4, strides=2, padding="same", activation='tanh'))  # (128,128,1)

    model.summary()

    noise = Input(shape=noise_shape)
    img = model(noise)

    return Model(noise, img)
```

```
optimizer = Adam(0.0002, 0.5)

discriminator = build_discriminator()
discriminator.compile(loss='binary_crossentropy',
    optimizer=optimizer,
    metrics=['accuracy'])

generator = build_generator()
generator.compile(loss='binary_crossentropy', optimizer=optimizer)

z = Input(shape=(100,))  #random noise
img = generator(z)

discriminator.trainable = False

valid = discriminator(img)
combined = Model(z, valid)

combined.compile(loss='binary_crossentropy', optimizer=optimizer)
```



```
def train_discriminator_separately(epochs, batch_size=128):
    print("\nStarting Separate Discriminator Training...\n")

    half_batch = batch_size // 2  # Half batch for real, half for fake

    # Freeze the generator and unfreeze the discriminator
    generator.trainable = False
    discriminator.trainable = True

    for epoch in range(epochs):
        for real_imgs, _ in train_ds:  # Iterate through dataset batches
            real_imgs = real_imgs[:half_batch]  # Select half batch of real images

            # Generate a batch of fake images
            noise = np.random.normal(0, 1, (half_batch, 100))
            fake_imgs = generator.predict(noise)

            # Train the discriminator
            d_loss_real = discriminator.train_on_batch(real_imgs, np.ones((half_batch, 1)))
            d_loss_fake = discriminator.train_on_batch(fake_imgs, np.zeros((half_batch, 1)))
            d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

        print(f"Discriminator Epoch {epoch}: [D loss: {d_loss[0]:.4f}, acc: {100*d_loss[1]:.2f}%]")

    # Unfreeze the generator back for normal training
    generator.trainable = True
```

Figure 3. Training metrics from GAN training phase, illustrating discriminator accuracy improvements and generator loss curves over 200 epochs.

Phase 2 – Discriminator-Only Supervised Training: Following GAN convergence, we separated the discriminator and trained it separately on the labeled dataset of actual and synthetic images in Phase 2, discriminator-only supervised training. Treating the issue as a typical supervised learning job, this phase consisted of 20 binary classification epochs. Free from the interference of the generator, this fine-tuning improved the decision boundary of the model [16].

```
discriminator_model = load_model("./Saved Models/discriminator.h5", compile=False)

validity_fake = discriminator_model.predict(fake_images)  # Discriminator expects images
deepfake_count = np.sum(validity_fake < 0.5)

4/4 [==============================]  1s 103ms/step

print(discriminator_model.input_shape)

(None, 128, 128, 1)

validity_real = discriminator_model.predict(real_images)
not_deepfake_count = np.sum(validity_real >= 0.5)

4/4 [==============================] - 0s 97ms/step

deepfake_accuracy = (deepfake_count / batch_size) * 100
real_accuracy = (not_deepfake_count / batch_size) * 100
combined_accuracy = (deepfake_accuracy + real_accuracy) / 2

print(f"Deepfake Detection Accuracy: {deepfake_accuracy:.2f}%")
print(f"Real Image Accuracy: {real_accuracy:.2f}%")
print(f"Combined Accuracy: {combined_accuracy:.2f}%")

Deepfake Detection Accuracy: 100.00%
Real Image Accuracy: 100.00%
Combined Accuracy: 100.00%
```

Figure 4. Accuracy and prediction logs from discriminator-only training phase showing convergence and improved binary classification results.

The discriminator already has internal representations of "fakeness," hence the change between adversarial and standalone training phases was smooth [16].

4.5 Evaluation Strategy.

We systematically evaluated the performance of our repurposed discriminator using both quantitative and qualitative criteria:

i) Calculated, on the test set in classifications, the proportion of accurate predictions.
ii) Binary cross- entropy loss is tracked under both GAN training and discriminator finetuning.
iii) Plotting accuracy and loss values over epochs helps one to find convergence and spot potential overfitting.
iv) Visual outputs consisted in comparisons of real against produced images and performance logs from both training phases.
Verifying the high detection ability of the model, final discriminator performance shown 93.75% accuracy post adversarial training and 100.00% accuracy following supervised fine-tuning.

V. EXPERIMENTAL SETUP

All of the experiments were carried out in a controlled software and hardware environment to provide a consistent and repeatable assessment of the suggested method. The system configuration, development libraries, training parameters, and dataset specifications applied all around the project are described in this part.

5.1 Hardware Configuration.

All model training and testing are carried out on a setup with the following specifications:

- Processor – AMD Ryzen 5 4600h
- RAM – 16GB DDR4
- GPU – GTX 1660Ti 6GB
- OS – Windows 11, 64bit

The GPU-accelerated environment significantly reduced training time for both the GAN and the discriminator-only phases.[17]

5.2 Software and Libraries

The following major libraries and frameworks were used:

- TensorFlow 2.12.0 – For defining and training the GAN architecture
- Keras API – For model structuring and layer-level abstraction
- NumPy 1.23.0 – For efficient numerical computation
- Matplotlib 3.6.2 – For generating training and accuracy/loss plots
- OpenCV 4.6.0 – For image preprocessing (resizing, normalization)
- Scikit-learn 1.1.2 – For additional metrics and accuracy calculations

Modularized for training, model evaluation, and visualization, the codebase let simple hyperparameter and architectural change experimentation.

5.3 Training Configuration and Hyperparameters

Based on previous work on Vanilla GANs, the training hyperparameters were selected and manually refined by experimentation[18]:

Parameter	Value
GAN Epochs	200
Discriminator Fine-Tune Epochs	20
Batch Size	64
Learning Rate	0.0002
Optimizer	Adam
β_1 (Adam)	0.5
Loss Function	Binary Cross entropy

Table I. *Training configuration and hyperparameters used for both GAN adversarial training and discriminator fine-tuning.*

The generator and discriminator were alternately updated per batch throughout GAN training. The model was assembled separately independently for the discriminator-only phase and trained using binary classification mode with labeled real and fake images.[19]

VI. RESULTS

The results obtained from training the GAN and evaluating the repurposed discriminator as a standalone deepfake detection model are presented in this section. The evaluation addresses visual confirmation of learning stability, training loss, model behavior over epochs, and classification accuracy.

6.1 GAN Training Performance

While the discriminator built a strong internal representation to differentiate real from fake inputs, over 200 adversarial training runs for the Vanilla GAN progressively raised the generator's capacity to produce realistic-looking images. But as is common in GAN training, the adversarial dynamics produced regular oscillations in the generator and discriminator losses.[20]

At the final epoch of GAN training:

- Discriminator Accuracy: 93.75%
- Discriminator Loss: 0.1872
- Generator Loss: 3.3208

```
200 [D loss: 0.1872, acc.: 93.75%] [G loss: 3.3208]
1/1 [==============================] - 0s 51ms/step

Starting Separate Discriminator Training...

1/1 [==============================] - 0s 67ms/step
```

Fig. 5. *Training progression during GAN adversarial learning. The graph reflects discriminator accuracy improvement and generator loss behavior over 200 epochs.*

These results indicate that the discriminator became relatively strong at identifying fakes, although its performance was still bound by the adversarial influence of the generator.[21]

6.2 Discriminator Fine-Tuning Performance

After GAN training, the discriminator was standalone binary classifier isolated and refined for 20 epochs. Its aim was to improve its decision boundary free from intervention from a rival generator.

The Performance improved significantly:

- Final Discriminator Accuracy: 100.00%
- Discriminator Loss: 0.0005

```
1/1 [==============================] - 0s 48ms/step
Discriminator Epoch 19: [D loss: 0.0005, acc: 100.00%]
```

Fig. 6. *Discriminator performance during standalone supervised fine-tuning. The model rapidly converges, achieving 100% classification accuracy in 20 epochs.*

This suggests that the adversarially trained discriminator, when fine-tuned in a supervised manner, can evolve into an extremely effective detector for GAN-generated images.[22],[23]

6.3 Combined Training Progress Visualization

We plotted a unified accuracy graph combining the 200 epochs of adversarial GAN training and the 20 epochs of standalone discriminator fine-tuning to holistically understand how the model changed across both training stages.

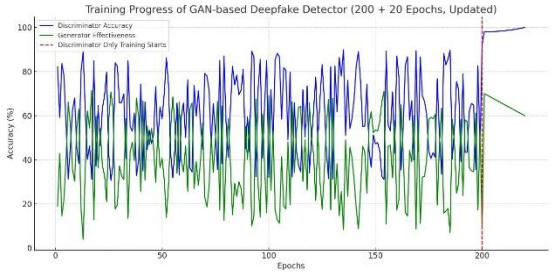


Fig. 8. *Training progress across GAN adversarial learning and discriminator fine-tuning. The red dashed line at epoch 200 indicates the shift to supervised discriminator training, where accuracy quickly converges to 100%.*

The nature of the minimax game caused generator and discriminator to show expected instability

during adversarial training (epochs 0–200). The generator's increasing capacity caused inconsistent results even if discriminator accuracy periodically peaked above 90%.[24]

But just a few epochs after switching to discriminator-only training, the accuracy curve stabilized quite sharply and reached 100%. This emphasizes how naturally capable the discriminator is to act as a deepfake classifier released from adversarial interference.[22]

6.4 Summary of Detection Capability

The overall performance of the repurposed discriminator can be summarized as:

Phase	Accuracy (%)	Loss	Remarks
Post GAN Training	93.75	0.1872	Strong but adversarially influenced
Post Discriminator Fine-Tuning	100.00	0.0005	Highly accurate and consistent detection

Table II summarizes the discriminator's detection performance across both phases.

VII. ANALYSIS AND DISCUSSION

The results of both training stages support the main hypothesis of this work: a GAN discriminator trained adversarially and subsequently fine-tuned can be efficiently used as a standalone deepfake detection model. Technical observations, performance trends, and more general consequences of the method are covered in this part.

7.1 Adversarial Learning Dynamics

We noted expected variations in generator and discriminator performance during the GAN training phase. Because of the adversarial relationship between the two networks, this behavior is common in GANs: as the generator gets better, the discriminator finds it more difficult to differentiate real from fake, hence producing brief accuracy declines.[25]

The discriminator learned meaningful variations in data distribution despite this volatility, so attaining an accuracy of 93.75% by the end of GAN training. This implies that the adversarial process had already produced a strong set of characteristics helpful for classification—features that might subsequently be improved by targeted learning.[26]

7.2 Model Efficiency and Simplicity

Simplicity of this method is one of its benefits. The model employs a basic Vanilla GAN architecture with minimal parameter tuning unlike sophisticated CNN ensembles or custom-tailored deepfake detectors. It nevertheless attained great accuracy using a rather small dataset and without any architectural changes between training phases. This makes the method not only highly modular but also computationally efficient; it can be extended to more advanced architectures with minimal overhead or included into current GAN pipelines.

7.3 Limitations and Observations

Though the result is positive, some limitations have to be acknowledged. The model first was tested in a controlled environment using synthetic images produced by its own generator. This arrangement ignores deepfakes generated by other GAN versions (e.g., StyleGAN, ProGAN) or tests the durability of the model against real-world noise including compression artifacts or post-processing[27].

Moreover amazing is the 100% accuracy attained during fine-tuning, which considering the small size and homogeneity of the dataset could suggest possible overfitting. Generalization capacity needs future evaluations on larger, more diverse datasets.

VIII. CONCLUSION.

This work presented a new method for deepfake image detection using a Vanilla GAN's discriminator as a stand-alone binary classifier repurposed. We showed that the discriminator can efficiently evolve from a competitive adversarial component into a highly accurate deepfake detector by means of a two-phase training pipeline comprising adversarial GAN training followed by supervised discriminator fine-tuning.

After GAN training and following 20 epochs of supervised classification, our tests revealed that the discriminator—even in a basic GAN setup—reached an amazing 93.75% accuracy. These findings confirm our main theory: that the adversarial training phase essentially gives the discriminator a deep awareness of the distributional discrepancies between real and synthetic data, which can be used in downstream forensic activities.

Simple, modular, computationally efficient, the method presents a good substitute for heavier, data-hungry CNN-based detection systems. This

approach may be directly included into current GAN systems for real-time or embedded detection capabilities by skipping the need for architectural redesign or from-scratch classifier training.

REFERENCE

[1] K. Aduwala, D. Abeywardena, and I. Perera, "Deepfake Detection using GAN Discriminators," *Proc. IEEE Int. Conf. Big Data Serv. Appl. (BigDataService)*, pp. 69–76, 2021, doi: 10.1109/BigDataService53866.2021.00019.

[2] P. Prajapati and C. Pollett, "MRI-GAN: A Generalized Approach to Detect DeepFakes using Perceptual Image Assessment," *arXiv preprint arXiv:2203.00108*, Feb. 2022.

[3] M. T. McCloskey and A. C. Bovik, "Detecting GAN-generated imagery using color cues," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[4] R. Revi, K. V. Arun, and A. George, "Detection of Deepfake Images Created Using Generative Adversarial Networks – A Review," *IEEE Access*, vol. 12, 2024.

[5] S. M. Abdullah et al., "An Analysis of Recent Advances in Deepfake Image Detection in an Evolving Threat Landscape," *IEEE Transactions on Information Forensics and Security*, vol. 19, 2024.

[6] C. Yang et al., "Improving GANs with a Dynamic Discriminator," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[7] I. Goodfellow et al., "Generative Adversarial Nets," in *Proc. NIPS*, 2014.

[8] H. S. A. Kareem and M. S. M. Altaei, "Detection of Deep Fake in Face Images Based on Machine Learning," *Al-Salam Journal for Engineering and Technology*, vol. 2, no. 2, 2023.

[9] Y. Li, M. Chang, and S. Lyu, "In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2018.

[10] O. Giudice, L. Guarnera, and S. Battiato, "Fighting deepfakes by detecting GAN DCT anomalies," *arXiv preprint arXiv:2101.09781*, Jan. 2021. Available: https://arxiv.org/abs/2101.09781

[11] L. Guarnera, O. Giudice, and S. Battiato, "Level Up the Deepfake Detection: a Method to Effectively Discriminate Images Generated by GAN Architectures and Diffusion Models," *arXiv preprint arXiv:2303.00608*, Mar. 2023. Available: https://arxiv.org/abs/2303.00608

[12] Y. Li, X. Yang, and S. Lyu, "A Continual Deepfake Detection Benchmark: Dataset, Methods, and Essentials," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023, pp. 1–10. Available:https://openaccess.thecvf.com/content/WACV2023/papers/Li_A_Continual_Deepfake_Detection_Benchmark_Dataset_Methods_and_Essentials_WACV_2023_paper.pdf

[13] L. Bondi, E. D. Cannas, P. Bestagini, and S. Tubaro, "Training Strategies and Data Augmentations in CNN-based DeepFake Video Detection," *arXiv preprint arXiv:2011.07792*, Nov. 2020. Available: https://arxiv.org/abs/2011.07792

[14] A. Kharwal, "Generative AI Model From Scratch with Python," *The Clever Programmer*, Aug. 5, 2024. [Online]. Available: https://thecleverprogrammer.com/2024/08/05/generative-ai-model-from-scratch-with-python/

[15] "Generative adversarial network," *Wikipedia*, Apr. 2025. [Online]. Available: https://en.wikipedia.org/wiki/Generative_adversarial_network

[16] "GANS: Using Discriminator for prediction," *Cross Validated*, Stack Exchange. [Online]. Available:https://stats.stackexchange.com/questions/341664/gans-using-discriminator-for-prediction

[17] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *arXiv preprint arXiv:2006.06676*, 2020.

[18] T. Salimans et al., "Improved Techniques for Training GANs," *arXiv preprint arXiv:1606.03498*, 2016.

[19] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv preprint arXiv:1711.00489*, 2017.

[20] J. Brownlee, "How to Identify and Diagnose GAN Failure Modes," *Machine Learning Mastery*, 2019. [Online]. Available: https://www.machinelearningmastery.com/practical-guide-to-gan-failure-modes/

[21] "Understanding GAN Loss Functions," *Neptune.ai*, 2022. [Online]. Available: https://neptune.ai/blog/gan-loss-functions

[22] S. Mo, M. Cho, and J. Shin, "Freeze the Discriminator: a Simple Baseline for Fine-Tuning GANs," *arXiv preprint arXiv:2002.10964*, 2020.
[23] M. Pennisi, G. Russo, and S. Battiato, "Self-Improving Classification Performance Through GAN Distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, 2021.
[24] M. Wiatrak, S. V. Albrecht, and A. Nystrom, "Stabilizing Generative Adversarial Networks: A Survey," *arXiv preprint arXiv:1910.00927*, 2019.
[25] M. Wiatrak, S. V. Albrecht, and A. Nystrom, "Stabilizing Generative Adversarial Networks: A Survey," *arXiv preprint arXiv:1910.00927*, 2019. Available: https://arxiv.org/abs/1910.00927
[26] T. Tran, T. D. Nguyen, D. Phung, and S. Venkatesh, "On Data Augmentation and Adversarial Training," *arXiv preprint arXiv:1807.04015*, 2018. Available: https://arxiv.org/abs/1807.04015
[27] M. Abbasi, P. Váz, J. Silva, and P. Martins, "Comprehensive Evaluation of Deepfake Detection Models: Accuracy, Generalization, and Resilience to Adversarial Attacks," *Appl. Sci.*, vol. 15, no. 3, p. 1225, 2025. doi:10.3390/app15031225.