Detecting GAN-Generated Face Deepfakes

Authors: Mattis Haumann, Sven Kurth, Sebastian Uedingslohmann
Submission Type: Final Exam Paper
Course: Machine Learning and Deep Learning
Degree: M.Sc. Business Administration and Data Science
Length: 13 pages (31,056 characters)
GitHub: https://github.com/seue24/GAN-generated-Face-Image-Detection

August 17, 2025

Abstract

This paper investigates the performance and limitations of machine and deep learning models in detecting GAN-generated face deepfakes. We created a balanced dataset of 16,000 face images, consisting of 8,000 real faces and 8,000 synthetic faces generated using four widely known GAN architectures: StyleGAN2, StyleGAN3, ProGAN, and StarGAN. To evaluate detection performance, we first trained an SVM with an RBF kernel, followed by a shallow CNN (2 convolutional layers) and a deeper CNN (4 convolutional layers). The deeper CNN emerged as the best-performing model, achieving an overall accuracy of 91% on the test set, and was selected for further evaluation. In our case, StarGAN images were particularly difficult to detect, whereas ProGAN images were correctly identified in almost all cases. Furthermore, the model failed to reliably detect face images from an unseen GAN type (ProjectedGAN), with accuracy dropping to 59%, highlighting the challenge of generalization in this domain. These findings underline both the promise and challenges of deepfake detection models, and the need for robust and generalizable models.

Keywords: Generative Adversarial Networks, Deepfake Detection, Face Image Classification, Convolutional Neural Networks, Support Vector Machines

1 Introduction

Generative Adversarial Networks (GANs) were first introduced by Goodfellow et al. in 2014 and have since become one of the most powerful and most used techniques for creating realistic synthetic media. Their ability to generate photorealistic outputs unlocked creative and commercial opportunities but also led to the rise of deepfakes. People are no longer able to reliably distinguish those deepfake images from human-generated images (Bray et al., 2023), which presents a serious challenge for digital platforms, especially social networks (Atlam et al., 2025), where online fraud is on the rise (Yang et al., 2024). Face deepfakes are particularly dangerous as they can be used for identity fraud, impersonation, and social engineering attacks. According to Sumsub's *Identity Fraud Report 2024*, the fraud rate on social networks has more than doubled–from 5.4%

in 2023 to 11.8% in 2024–largely driven by fake profiles and the increasing accessibility of deepfake tools (Sumsub, 2024).

In this paper, we investigate the ability of machine learning models to distinguish GAN-generated face images from real ones.

2 Motivation and Research Questions

The increasing realism of GAN-generated synthetic images has created an urgent need for robust and generalizable deepfake detection systems. As new generative models continue to emerge, detectors must not only perform well on known deepfakes but also adapt to new or unseen GAN types.

Our motivation lies in building a model that achieves reliable classification performance on GAN-generated face deepfakes using a dataset of real and synthetic images, individually curated by us. A key use case for this model is its integration into social media platforms, where it could be used as an automatic authentication layer to help platforms to detect fake profiles and protect users from impersonation and fraud. The model could then also be continuously improved by training it on new types of deepfakes as they appear.

We aim to explore the following main research question and two sub-questions:

- **Research Question (RQ):** *Can machine learning models distinguish GAN-generated face images from real face images?*

- **Sub-Research Question 1 (SR1):** *Which GAN-generated face images are most often misclassified as real by the model?*
- **Sub-Research Question 2 (SR2):** *How does the model perform on face images generated by an unseen GAN-type?*

3 Related Work

Deepfake detection has become a widely researched topic in computer vision and machine learning, covering manipulated videos, audio, and images produced by a variety of different techniques. Within this broad landscape, our focus is on detecting GAN-generated face images. One of the earliest studies in this domain was conducted by Do et al. (2018), who used pretrained convolutional neural networks (CNNs) to distinguish real faces from those generated by early GAN models such as DC-GAN and PG-GAN. They tested different pretrained models, including VGG16 and ResNet50, and reported classification accuracies ranging from approximately 73% to 80%.

From there on, deep learning-based techniques, specifically CNNs, were widely used in this domain. More recently, several studies have proposed lightweight CNN architectures to reduce model complexity while keeping good model performance. For example, a simple CNN optimized for low-resolution deepfake image detection reduced parameter requirements by 92% while preserving accuracy, making it suitable for mobile applications (Sabareshwar et al., 2024). Similarly, Arshad et al. (2024) introduced a CNN only consisting of two convolutional layers, two pooling layers and one dense layer achieving an accuracy of 95%.

In parallel, traditional machine learning methods have also been explored. For instance, Rafique et al. (2023) proposed a hybrid approach combining CNN feature extraction with Support Vector Machines (SVM) classification, achieving 88.6% accuracy.

In addition to CNN-based approaches, other new studies have begun exploring transformer-based architectures. For example, Arshed et al. (2023) introduced a Vision Transformer (ViT) model which was able to achieve a remarkably high detection accuracy of up to 99.95%.

However, a consistent limitation identified across the literature is the limited generalization capability of these models. While they tend to perform well when trained and tested on the same GAN-type, they tend to do worse when applied to previously unseen GANs (Arshed et al., 2023).

Despite the variety of existing research, we found no study to date that evaluated model performance on a dataset combining our specific selection of both older and newer GAN-generated face images, making it interesting to observe how our model generalizes across this unique and diverse dataset.

4 Conceptual Framework

The conceptual framework guiding this study is outlined in Figure 1 and consists of five sequential stages. The process begins with the collection of real and synthetic face images from publicly available datasets, including CelebA and FFHQ for real images, and ProGAN, StyleGAN2/3, ProjectedGAN and StarGAN for fake images. The data is then pre-processed through filtering, normalization, class rebalancing, and splitting. Note that images generated by ProjectedGAN were withheld from training data to be used exclusively for evaluating the model's generalization ability. For the modelling stage, three approaches are applied: an SVM, a shallow CNN, and a deeper CNN. Evaluation I focuses on comparing the different models on test data, whereas Evaluation II tests the best model's generalization on previously unseen ProjectedGAN samples.

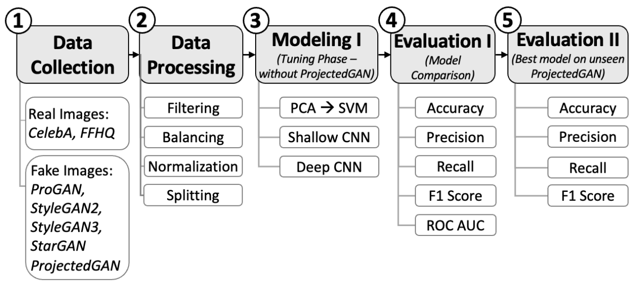


Figure 1. Overview of the conceptual framework.

5 Methodology

5.1 Data Description and Analysis

Our study used a combination of real and GAN-generated face image datasets. The real images originated from the popular FFHQ and CelebA datasets. FFHQ includes high-quality, high-resolution face images with diverse age, ethnicity, and facial expressions. Similarly, CelebA contributed with celebrity images exhibiting diverse facial attributes and pose variations.

The GAN-generated face images span a diverse range of GAN architectures from different years. We collected face images generated by StyleGAN2 from NVIDIA (Alarcon, 2020), StyleGAN3 from NVIDIA (Kakkar, 2021), ProGAN from Karras et al. (2018), StarGAN from Choi et al. (2018), and ProjectedGAN from Sauer et al. (2021).

The data was compiled from multiple sources. We incorporated the Kaggle-hosted version of CelebA[1]. From the ArtiFace dataset (Khan & Valles, 2024), we obtained real face images from FFHQ, along with GAN-generated face images from ProjectedGAN and StarGAN. Synthetic face images from StyleGAN2, StyleGAN3, and ProGAN were included from the Kaggle dataset fake-face-images-generated-from-different-gans[2]. Our initial data was characterized by a moderate class imbalance between real and fake face images, as illustrated in Table 1.

Table 1
Number and percentage of sourced images grouped by real and fake labels.

Label	Origin	Count	Percentage
Real	CelebA	202,599	72.19%
	FFHQ	49,255	17.55%
Fake	StyleGAN 2	7,512	2.66%
	StyleGAN 3	7,100	2.51%
	StarGAN	7,100	2.51%
	ProGAN	7,100	2.51%
	ProjectedGAN	1,641	0.58%

Figure 2 displays random samples from both the real image sources and the various fake face images from different GANs. Notably, the visual realism of the fake images varies across GAN architectures. Images produced by the more modern GAN architectures StyleGAN2 and StyleGAN3, followed by ProjectedGAN, are often nearly indistinguishable from real face images. In contrast, images generated by the less modern GANs StarGAN and ProGAN tend to exhibit more noticeable artifacts, such as blurring and unusual brightness, making them easier to identify as synthetic upon visual inspection.

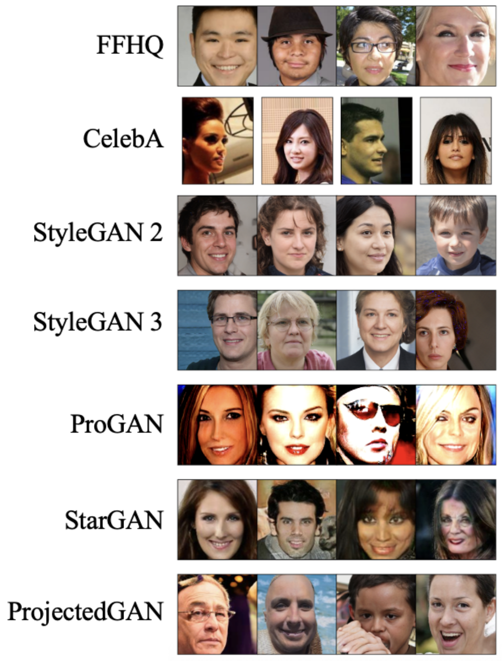


Figure 2. Example images from each sublabel, including both real and synthetic GAN-generated faces.

Apart from visual differences, we observed notable inconsistencies in the image formats across the data. Specifically, image resolutions varied, with widths and heights ranging from 128 to 256 pixels. Moreover, CelebA images are portrait-oriented with an average aspect ratio of approximately 0.82, whereas all other images follow a square format with an average aspect ratio of 1.0. These format differences highlighted the need for preprocessing to ensure consistency

[1] https://www.kaggle.com/datasets/jessicali9530/celeba-dataset
[2] https://www.kaggle.com/datasets/sambit9238/fake-face-images-generated-from-different-gans

and avoid introducing format-related biases during model training.

5.2 Data Preprocessing

5.2.1 Data Filtering

To prevent bias by over-representing certain patterns in the data, we systematically checked the data for duplicate images and removed them. Duplicate detection was performed using the approach proposed by Li et al. 2015, which involves computing the perceptual hash of each image and identifying duplicates based on hash similarity. Through this process, we identified and removed a total of 963 duplicate images. Furthermore, we removed 53 images with average RGB values below 5 or above 250 to eliminate extreme outliers in color intensity. This reduced potential bias from abnormal inputs.

5.2.2 Data Balancing

To achieve an efficient training process while maintaining sufficient data for robust learning and evaluation, the dataset size was set to 16,000 images. To ensure that the models effectively learn the distinguishing features between real and GAN-generated face images, we balanced the dataset by equalizing the number of real and fake samples. Additionally, the real class was composed of an equal number of images from FFHQ and CelebA to avoid source-specific bias within the real class. Furthermore, we ensured that the fake class contained an equal distribution of the various GAN-generated image categories to prevent the model from overfitting to artifacts specific to individual GAN types and to improve generalization. Notably, we excluded the ProjectedGAN category to reserve it for testing the model's generalization ability to an unseen type of GAN-generated face images.

Table 2
Final dataset distribution of real and fake samples by origin, including class totals.

Class	Origin	Count	Total
Real	FFHQ	4,000	8,000
	CelebA	4,000	
Fake	StyleGAN2	2,000	8,000
	StyleGAN3	2,000	
	ProGAN	2,000	
	StarGAN	2,000	
Total			**16,000**

5.2.3 Data Normalization

To remove format-related bias, we applied a center crop to each image, followed by resizing to a standardized 128×128 square format. Center cropping eliminated aspect-ratio-related bias and ensured that facial regions remained the focus. Moreover, standardizing image dimensions also mitigated the risk of models associating specific resolutions with class labels. The 128×128 resolution was chosen as it aligned with the minimum dimensions observed in the data set. Each resized image was also flattened into a one-dimensional feature vector, a necessary transformation to ensure compatibility with models such as SVMs.

5.2.4 Dataset Split

For the training and evaluation of our models, we deployed a 70% training set, 15% validation set, and 15% test set split. Including a separate test set provided a further unbiased evaluation of the model's generalization ability.

5.3 Modeling, Methods, and Tools

To address the problem of GAN-based deepfake detection, this study creates a SVM and two CNNs. CNNs were chosen as they are specifically designed for pattern recognition within images (O'Shea & Nash, 2015), and they are by far the most commonly used architecture in face fake image detection (Arshad et al., 2024; Do et al., 2018; Sabareshwar et al., 2024). SVM was used as a baseline due to its established role in traditional machine learning and its frequent application in deepfake face detection tasks (Rana et al., 2022).

5.3.1 Support Vector Machine

The choice of SVM was informed by its wide use in image classification tasks and its simplicity in implementation. Based on related research, SVM has shown better performance compared to other established classifiers such as Random Forest and Logistic Regression when applied to learning visual features (Saberioon et al., 2018; Zhang et al., 2016).

Before training, we applied a Principal Component Analysis (PCA) to reduce the dimensions from 49,152 to 216 principal components, which accounted for 90% of the variance. This improved computational efficiency, reduced the risk of overfitting, and enhanced the SVM model's ability to handle more complex data structures (Géron, 2019).

The model was trained using the `SVC` class from `sklearn`, and a grid search was conducted to optimize the kernel type, the regularization parameter C, and the kernel coefficient γ. For the values of C, we followed the range proposed by Hsu et al. (2003), which is commonly used in SVM hyperparameter tuning. The grid search explored both RBF and polynomial kernels using 5-fold cross-validation. The best-performing configuration was an RBF kernel with $C = 2$ and $\gamma = 3.05 \times 10^{-5}$. These results suggest that the RBF kernel was better suited to the PCA-transformed data, and the selected model served as a strong baseline for comparison with the CNN architectures.

5.3.2 Convolutional Neural Networks

Two variations of CNNs were implemented for this study. The first, referred to as $\text{CNN}_{\text{shallow}}$, featured a simplified architecture with two convolutional layers. The second, denoted as CNN_{deep}[3] in our study, incorporated two additional convolutional layers, meaning four in total.

Both CNN architectures employed ReLU activation functions in all convolutional layers to mitigate the vanishing gradient problem during backpropagation (Géron, 2019). As no signs of dying ReLUs were observed during training, alternative activation functions such as Leaky ReLU were not deemed necessary. Each convolutional layer was followed by batch normalization to stabilize and facilitate faster training (Santurkar et al., 2018), along with max pooling to reduce spatial dimensions and retain dominant features. The final part of both architectures comprises a flatten layer, a fully connected dense layer

[3] Note that *deep* here indicates only that this network is deeper compared to $\text{CNN}_{\text{shallow}}$, not necessarily that it qualifies as a deep CNN architecture per se.

with ReLU activation, a dropout layer for regularization, and a sigmoid-activated output layer suitable for binary classification. As the two models differ primarily in architectural depth, this design facilitates an evaluation of whether increased depth improves classification performance, as often suggested in literature (Lokner Lađević et al., 2024). During training and hyperparameter tuning, CNN$_{\text{deep}}$ exhibited a stronger tendency to overfit. To counter this, L2 regularization was applied to all convolutional layers of CNN$_{\text{deep}}$. The complete architectures are summarized in Table 3.

In addition to architectural design choices, the hyperparameters and training configurations of the CNNs were tuned in several stages. Key parameters included the number of epochs, batch size, learning rate, dropout rate, and kernel size. Initial tuning was performed through manual trial and error to understand the models' sensitivity to various settings, followed by a systematic grid search over learning rate, batch size, dropout rate, and kernel rate (see Appendix A). The models were trained using binary cross-entropy loss, the standard choice for binary classification, and trained for up to 25 epochs at first with early stopping (patience = 2) and `restore_best_weights` enabled to retain the best-performing parameters.

Table 3
Architectures of CNN$_{\text{shallow}}$ and CNN$_{\text{deep}}$.

(a) CNN$_{\text{shallow}}$ Architecture

Layer	Type	Output Shape	Parameters
Conv1	Conv2D (ReLU)	(126, 126, 32)	896
BatchNorm1	BatchNormalization	(126, 126, 32)	128
Pool1	MaxPooling2D (2x2)	(63, 63, 32)	0
Conv2	Conv2D (ReLU)	(61, 61, 64)	18,496
BatchNorm2	BatchNormalization	(61, 61, 64)	256
Pool2	MaxPooling2D (2x2)	(30, 30, 64)	0
Flatten	Flatten	(57600)	0
Dense1	Dense (ReLU)	(64)	3,686,464
Dropout	Dropout (0.4)	(64)	0
Dense2	Dense (Sigmoid)	(1)	65

(b) CNN$_{\text{deep}}$ Architecture

Layer	Type	Output Shape	Parameters
Conv1	Conv2D (ReLU, L2)	(126, 126, 32)	896
BatchNorm1	BatchNormalization	(126, 126, 32)	128
Pool1	MaxPooling2D (2x2)	(63, 63, 32)	0
Conv2	Conv2D (ReLU, L2)	(61, 61, 64)	18,496
BatchNorm2	BatchNormalization	(61, 61, 64)	256
Pool2	MaxPooling2D (2x2)	(30, 30, 64)	0
Conv3	Conv2D (ReLU, L2)	(28, 28, 128)	0
BatchNorm3	BatchNormalization	(28, 28, 128)	512
Pool3	MaxPooling2D (2x2)	(14, 14, 128)	0
Conv4	Conv2D (ReLU, L2)	(12, 12, 256)	295,168
BatchNorm4	BatchNormalization	(12, 12, 256)	1,024
Pool4	MaxPooling2D	(6, 6, 256)	0
Flatten	Flatten	(9216)	0
Dense1	Dense (ReLU)	(256)	2,359,552
Dropout	Dropout (0.4)	(256)	0
Dense2	Dense (Sigmoid)	(1)	257

Final hyperparameter selections were based on their contribution to training stability and resistance to overfitting. A learning rate of 0.00001 proved most influential. A kernel size of (3,3), dropout rate of 0.4, and batch size of 32 for both models proved to be an effective configuration. In the ultimate model setup, the number of epochs was increased to 30, as in particular CNN$_{\text{deep}}$ had not yet converged within its initial training window.

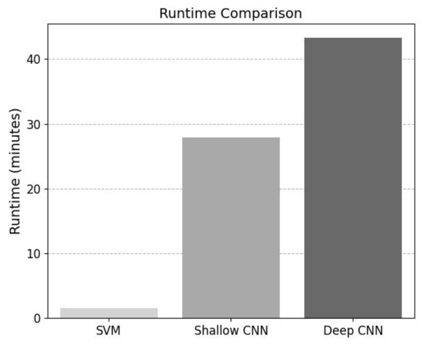
To improve generalization, data augmentation was applied during all training runs using Keras' `ImageDataGenerator`, incorporating transformations such as random zooming (up to 20%) and brightness adjustments (ranging from 70% to 130% of the original). These augmentations enhanced the model's robustness to common image variations encountered in real-world scenarios.

5.4 Model Complexity Analysis

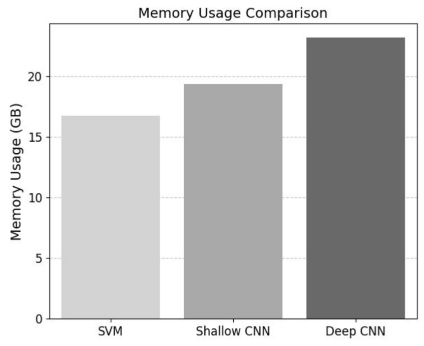
While evaluation metrics such as accuracy and F1-score are essential for assessing model performance, they must be weighed against model complexity and computational efficiency. In our study, all models were benchmarked on a high-performance 32-core Intel Xeon Gold 6130 CPU with 384 GB RAM. As shown in Figure 3a, we observed substantial differences in training times between the models.

The SVM required the least training time, taking only a bit less than two minutes. Despite using an RBF kernel, its relatively low training cost is due to the simplicity of the feature space and the limited dataset size, which kept kernel computations and matrix operations manageable. It also had the lowest memory usage among all models, consuming 16.42 GB during training (Figure 3b).

In contrast, the CNNs required significantly more computational resources. $\text{CNN}_{\text{shallow}}$ represented a middle ground in terms of runtime and memory usage (29 minutes and 18.56 GB), while CNN_{deep} demanded the most (44 minutes and 22.57 GB). Each additional convolutional layer increases the number of filters and parameters, resulting in more operations and a longer backpropagation process. Moreover, the difference in runtime was also influenced by the number of training epochs, which varied due to early stopping and was itself affected by the greater architectural depth of CNN_{deep}.



(a) Runtime comparison across models



(b) Memory usage comparison across models

Figure 3. Model complexity analysis.

While $\text{CNN}_{\text{shallow}}$ trains faster due to its simplicity, CNN_{deep} comes with higher computational cost. Whether this added complexity is justified remains unclear without considering performance, pointing to the common trade-off between efficiency and potential accuracy gains. Thus is particularly relevant for sensitive applications like deepfake face detection.

5.5 Evaluation Metrics

To evaluate and compare the models performance, we report accuracy, precision, recall, F1 score, and ROC AUC. These metrics have been widely adopted in previous deepfake detection research and provide a comprehensive assessment of model performance (Ben Aissa et al., 2024; Heidari et al., 2024; Wang et al., 2023).

Accuracy measures the general correctness of the predictions and is appropriate given that we used a balanced test dataset. Precision and recall capture how the model handles false positives and false negatives, both of which are critical in our security-sensitive case. False positives (misclassifying real faces as fakes) can lead to undesired account removal, while false negatives (failing to detect actual fakes faces) can enable harm through identity fraud and impersonation. To balance both error types, we include the F1 score, which provides insight into the model's ability to distinguish between real and fake images regardless of the classification threshold.

Rather than prioritizing a single metric, we evaluate performance holistically, recognizing that different metrics capture distinct aspects of model effectiveness. This approach reflects the dual risks of over- and under-detection in real-world systems such as social media moderation. The formulas for all metrics used are defined in Equations (1)–(4) below.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\textbf{Precision} = \frac{TP}{TP + FP}, \quad \textbf{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\textbf{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$\textbf{ROC AUC} = \int \text{TPR}\, d(\text{FPR}), \tag{4}$$

where TPR $= \frac{TP}{TP+FN}$ is the True Positive Rate and FPR $= \frac{FP}{FP+TN}$ is the False Positive Rate.

Abbreviations: TP = True Positives, FP = False Positives, TN = True Negatives, FN = False Negatives

6 Results

To address our research questions, the results are presented in three parts.

First, we compare model performance using selected evaluation metrics to assess how effectively each model detects GAN-generated face images.

Second, we examine which types of GANs pose the greatest challenge for the best-performing model.

Third, we evaluate this model on an unseen GAN type (ProjectedGAN) to draw conclusions about its generalization capability.

6.1 Model Comparison

Three models were evaluated on a test dataset consisting of 1,200 fake and 1,200 real face images. This balance was chosen on purpose as it enabled a more reliable interpretation of the evaluation metrics across both classes. Table 4 presents the classification performance of the different models on the test data.

The baseline SVM model achieved solid accuracy and an F1 score of 0.83, demonstrating strong baseline performance despite its simplicity. Notably, the SVM showed balanced precision scores across fake and real images, both at 0.83.

Both CNN architectures significantly outperformed the baseline. CNN$_{\text{shallow}}$ achieved a high accuracy and F1-score of 0.89, indicating its effectiveness in capturing subtle visual differences between fake and real images through low-level spatial feature extraction. CNN$_{\text{deep}}$ demonstrated further improvements, reaching an accuracy and F1-score of 0.91, and the highest ROC AUC of 0.97. While it is difficult to isolate and assess the precise contribution of the increased architectural depth, the results support the notion that the more effective extraction of complex and abstract features of the deeper CNN architecture contributed to the improved classification performance.

Training curves for both CNN models (Appendix B) reveal stable convergence patterns in validation accuracy and loss, confirming that both architectures achieved near-optimal performance under their given configurations.

CNN$_{\text{deep}}$ outperformed the other models across every metric. Considering this overall superior performance and the critical nature of deepfake detection tasks, the increased computational demand of CNN$_{\text{deep}}$ was deemed justified by us, leading to its selection as the best model for subsequent analyses.

Table 4
Per-class evaluation metrics and overall scores for all models.

Model	Class	Precision	Recall	F1 Score	Accuracy	ROC AUC
SVM	Fake	0.83	0.82	0.83	0.83	0.90
	Real	0.83	0.83	0.83		
CNN$_{\text{shallow}}$	Fake	0.89	0.89	0.89	0.89	0.96
	Real	0.89	0.89	0.89		
CNN$_{\text{deep}}$	Fake	0.93	0.89	0.91	0.91	0.97
	Real	0.90	0.93	0.91		

6.2 Error Analysis

While CNN$_{\text{deep}}$ performed well on some GAN types, such as ProGAN and StyleGAN2, it demonstrated increased misclassification (reduced accuracy) on others, most notably StarGAN. Figure 4 displays examples of the different GAN-types misclassified as real, and Table 5 shows the corresponding misclassification rates. Notably, while ProGAN samples were consistently detected as fake, StarGAN images proved more difficult to classify correctly, even though they appear visually less realistic than the more advanced NVIDIA StyleGAN series.

Overall, these findings suggest that the model's detection performance is not consistent across different generative models which highlights limitations in its generalization ability. They further imply that the visual cues learned by the CNN to identify fake images may differ between GAN types.

The results indicate that there is still room for improvement in the model's generalization ability, for example, by further training with an imbalanced dataset that emphasizes the more challenging GAN type.

Table 5
Fake samples misclassified as real by CNN$_{\text{deep}}$, by GAN type.

GAN Type	Misclassification Rate (%)
ProGAN	0.34
StyleGAN2	4.45
StyleGAN3	11.90
StarGAN	25.17

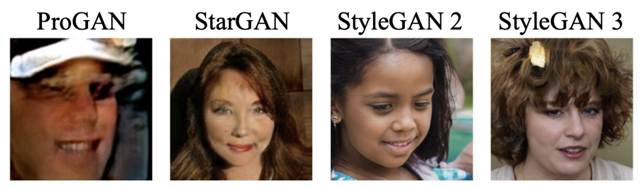


Figure 4. GAN-generated face images misclassified as real, by GAN type.

6.3 Evaluation on Unseen GAN

To draw further implications about the generalizability of our CNN_{deep}, we tested the model on another test set consisting of 1641 ProjectedGAN fake face images and 1640 real face images. Table 6 illustrates a selection of evaluation metrics results.

The results in Table 6 reveal a clear performance imbalance when evaluating the model on the unseen ProjectedGAN samples.

While the model maintains high recall for real images (0.93), it performs poorly in detecting fake face images, with a recall of only 0.14 and an F1-score of 0.24. This suggests that the model fails to identify the majority of fake images from ProjectedGAN, despite previously strong performance on the GAN types it was trained on. The precision for the fake class (0.66) indicates that when the model does predict an image as fake, it is often correct, but it rarely makes that prediction. The overall accuracy stands at just 0.59, which is only slightly better than random chance.

Table 6
Performance of CNN_{deep} on unseen ProjectedGAN samples.

Class	Precision	Recall	F1 Score	Accuracy
Fake (ProjectedGAN)	0.66	0.14	0.24	0.59
Real	0.52	0.93	0.67	

These findings highlight a lack of generalization of our model to unseen GAN types and a weakness in its ability to adapt to face images generated by an unfamiliar GAN type.

7 Discussion

7.1 Answers to Research Questions

Based on our findings, the main Research Question (**RQ**), whether machine learning models can reliably distinguish GAN-generated facial images from real facial images, can be answered with a *yes* but only if the model was trained on those type of fake facial images. This directly leads to **SR2**, which examined how well the model performs on facial images generated by an unseen GAN type. Here, the model struggled with previously unseen ProjectedGAN images, highlighting the challenge of generalizing detection capabilities to new generative methods.

Regarding **SR1**, which investigates which GAN-generated face images are most frequently misclassified as real by the model, results indicated that StarGAN samples posed particular detection challenges, whereas ProGAN images were mostly correctly identified as fake.

7.2 Implications

Our study identified important implications for the development of deepfake detectors in real-world applications.

We demonstrated that models can effectively classify certain fake face images as distinct from real ones. Already a simple CNN architecture can excel in detection, given it has been trained on the same GAN architecture.

Moreover, it showed key challenges regarding the generalizability of these models. Consistent with prior research, our CNN_{deep} struggled to classify images generated by a previously unseen GAN type. Furthermore, GAN artefacts seem to differ from each other in such a way that features learned from one type do not necessarily transfer well to others, further highlighting the limitations of model generalization.

This supports the practical need for the continuous identification and collection of new types of fake images, and for ensuring that detection models are trained accordingly to develop reliable detectors.

However, maintaining such an adaptability may be difficult to achieve in practice.

7.3 Limitations

This study has several limitations. The most recent GAN model included was released in 2021, which restricts our ability to draw conclusions about the model complexity required for the detection of more modern GAN architectures. Moreover, the training dataset comprised only four GAN types and was tested on a single unseen type (ProjectedGAN), limiting the generalizability of our results. It is possible that the model might perform better on other unseen GAN types or that ProjectedGAN happens to be particularly challenging and complex.

The dataset was also reduced to 16,000 standardized 128×128 images. While this helped make training more efficient, it may have suppressed subtle artifacts specific to certain GANs and introduced some risk of sampling bias, as the full diversity of the original data was not preserved.

Additionally, even though CelebA and FFHQ were chosen for their availability and demographic coverage, they are still curated datasets. Using real-world images, such as those found on social media, might have better reflected the actual conditions in which deepfake detectors are deployed.

Moreover, while the CNNs performed well overall, their decision-making process remains unclear. Since CNNs are often considered black-box models (Szandała, 2023), we cannot say for certain whether they are learning to detect actual GAN-specific artifacts or picking up on other patterns and biases present in the training data.

Lastly, a key limitation of this study is the uncertainty regarding the extent to which increased architectural depth contributed to the observed performance improvement, as the gains were minimal and not conclusively attributable to deeper feature extraction alone.

7.4 Ethical Considerations

Although intended to enhance online security, the use of machine learning models for deepfake detection raises important concerns about bias, fairness, and data protection.

In particular for CNNs, the decision-making processes remain intransparent. It should be considered that these models can exhibit biased patterns learned from training data, which can lead to unfair outcomes, especially if certain demographic groups are underrepresented or overrepresented. For example, if certain skin tones or facial features are underrepresented, the model may become less accurate at detecting deepfakes involving those individuals. To address this, we used both the widely adapted CelebA and the FFHQ dataset due to their broad availability and demographic diversity. However, studies have shown that the FFHQ dataset has a racial imbalance (Maluleke et al., 2022), potentially leading to bias in our model.

While using web-scraped real social media images to train the models could improve demographic balance, it would also introduce serious data privacy concerns as such data is classified as special category personal data under Article 9 (1) of the GDPR and is subject to strict processing conditions.

Deepfake detection is of increasing importance to protect individuals from the growing deception and misuse, however, such models must be developed and deployed with careful attention to potential biases and data privacy risks to ensure they do not cause more harm than they prevent.

8 Conclusion and Future Work

In this study, we trained three different models for GAN-generated deepfake face detection. The CNN with an architecture consisting of four convolutional layers performed best on this task, achieving an accuracy of 0.91. However, this model showed a performance imbalance across the GAN architectures it was trained and tested on, revealing limitations in generalizability. While the model performed well on fake face images generated by the same GAN architecture it was trained on, it struggled significantly with fake images from the previously unseen GAN type ProjectedGAN, achieving an accuracy of only 0.59.

To further improve model performance and generalizability, future research could explore several promising directions. First, analyzing the impact of using larger datasets on performance and generalizability may yield valuable insights.

Second, for a real-world fake detection model to achieve better generalization, more GAN types as well as other modern fake image types (e.g., Stable Diffusion or DALL·E) need to be included in the training data.

Third, more model configurations and hyperparameter combinations could be investigated.

Fourth, training models on higher image resolutions and examining the effects could also help determine whether standardizing images to 128×128 pixels removed important GAN artifacts and limited the models' ability to generalize.

Overall, a significant limitation of the current approach is that it remains unclear whether the CNNs classify images based on meaningful features or on artifacts that are inherent to real versus fake images. Building on this study, future work should explore explainable AI (XAI) methods to identify the features that models rely on when detecting deepfakes. As increasingly diverse and sophisticated deepfakes continue to emerge, future work in this domain will remain essential. Detection models must be continuously updated and re-trained to keep pace with evolving generative techniques.

References

Alarcon, N. (2020, June). Synthesizing High-Resolution Images with StyleGAN2. Retrieved May 15, 2025, from https://developer.nvidia.com/blog/synthesizing-high-resolution-images-with-stylegan2/

Arshad, T., Khan, M. H., & Farid, M. S. (2024). An efficient framework to recognize deepfake faces using a light-weight cnn. *Proceedings of the 2024 9th International Conference on Multimedia Systems and Signal Processing (ICMSSP)*, 24–29. https://doi.org/10.1145/3690063.3690064

Arshed, M. A., Alwadain, A., Faizan Ali, R., Mumtaz, S., Ibrahim, M., & Muneer, A. (2023). Unmasking deception: Empowering deepfake detection with vision transformer network. *Mathematics*, *11*(17), 3710. https://doi.org/10.3390/math11173710

Atlam, E.-S., Almaliki, M., Elmarhomy, G., Almars, A. M., Elsiddieg, A. M., & ElAgamy, R. (2025). SLM-DFS: A systematic literature map of deepfake spread on social media. *Alexandria Engineering Journal*, *111*, 446–455. https://doi.org/10.1016/j.aej.2024.10.076

Ben Aissa, F., Hamdi, M., Zaied, M., & Mejdoub, M. (2024). An overview of gan-deepfakes detection: Proposal, improvement, and evaluation. *Multimedia Tools and Applications*, *83*(11), 32343–32365.

Bray, S. D., Johnson, S. D., & Kleinberg, B. (2023). Testing human ability to detect âdeepfakeâ images of human faces. *Journal of Cybersecurity*, *9*(1), tyad011. https://doi.org/10.1093/cybsec/tyad011

Choi, Y., Choi, M., Kim, M., Ha, J. W., Kim, S., & Choo, J. (2018). StarGAN: 31st Meeting of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018 [Publisher: IEEE Computer Society]. *Proceedings - 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 8789–8797. https://doi.org/10.1109/CVPR.2018.00916
Funding Information:This work was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (No. NRF2016R1C1B2015924).Publisher Copyright:© 2018 IEEE.

Do, N.-T., Na, I., & Kim, S. (2018, October). *Forensics face detection from gans using convolutional neural network.*

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems* (2nd). O'Reilly Media.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems*, *3*. https://doi.org/10.1145/3422622

Heidari, A., Jafari Navimipour, N., Dag, H., & Unal, M. (2024). Deepfake detection using deep learning methods: A systematic and comprehensive review. *WIREs Data Mining and Knowledge Discovery*, *14*(2), e1520. https://doi.org/https://doi.org/10.1002/widm.1520

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2003). *A practical guide to support vector classifi-*

cation (tech. rep. No. 1396-1400). Department of Computer Science, National Taiwan University. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Kakkar, S. (2021, October). NVIDIA AI Releases StyleGAN3: Alias-Free Generative Adversarial Networks. Retrieved May 15, 2025, from https://www.marktechpost.com/2021/10/12/nvidia-ai-releases-stylegan3-alias-free-generative-adversarial-networks/

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018, February). Progressive Growing of GANs for Improved Quality, Stability, and Variation [arXiv:1710.10196 [cs] version: 3]. https://doi.org/10.48550/arXiv.1710.10196
Comment: Final ICLR 2018 version.

Khan, S. A., & Valles, D. (2024). Deepfake detection using transfer learning. *2024 IEEE 15th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 556–562. https://doi.org/10.1109/UEMCON62879.2024.10754706

Li, X., Li, J., & Huang, F. (2015). A secure cloud storage system supporting privacy-preserving fuzzy deduplication. *Soft Computing*, *20*. https://doi.org/10.1007/s00500-015-1596-6

Lokner Lađević, A., Kramberger, T., Kramberger, R., & Vlahek, D. (2024). Detection of ai-generated synthetic images with a lightweight cnn. *AI*, *5*(3), 1575–1593. https://doi.org/10.3390/ai5030076

Maluleke, V. H., Thakkar, N., Brooks, T., Weber, E., Darrell, T., Efros, A. A., Kanazawa, A., & Guillory, D. (2022). Studying bias in gans through the lens of race. https://arxiv.org/abs/2209.02836

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. https://arxiv.org/abs/1511.08458

Rafique, R., Gantassi, R., Amin, R., et al. (2023). Deep fake detection and classification using error-level analysis and deep learning. *Scientific Reports*, *13*, 7422. https://doi.org/10.1038/s41598-023-34629-3

Rana, M. S., Nobi, M. N., Murali, B., & Sung, A. H. (2022). Deepfake detection: A systematic literature review. *IEEE Access*, *10*, 25494–25513. https://doi.org/10.1109/ACCESS.2022.3154404

Sabareshwar, D., Raghul, S., Varalakshmi, M., Peer Mohamed, P., et al. (2024). A lightweight cnn for efficient deepfake detection of low-resolution images in frequency domain. *2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE)*, 1–6.

Saberioon, M., Císař, P., Labbé, L., Souček, P., Pelissier, P., & Kerneis, T. (2018). Comparative performance analysis of support vector machine, random forest, logistic regression and k-nearest neighbours in rainbow trout (oncorhynchus mykiss) classification using image-based features. *Sensors*, *18*(4), 1027. https://doi.org/10.3390/s18041027

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett

(Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. https : / / proceedings . neurips . cc / paper _ files / paper / 2018 / file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf

Sauer, A., Chitta, K., Müller, J., & Geiger, A. (2021, November). Projected GANs Converge Faster [arXiv:2111.01007 [cs]]. https://doi.org/10.48550/arXiv.2111.01007
Comment: To appear in NeurIPS 2021. Project Page: https://sites.google.com/view/projected-gan/.

Sumsub. (2024). *Identity fraud report 2024*. Sumsub. Retrieved May 10, 2025, from https://sumsub.com/fraud-report-2024/

Szandała, T. (2023). Unlocking the black box of cnns: Visualising the decision-making process with prism. *Inf. Sci.*, *642*(100). https://doi.org/10.1016/j.ins.2023.119162

Wang, X., Guo, H., Hu, S., Chang, M.-C., & Lyu, S. (2023). Gan-generated faces detection: A survey and new perspectives. *ECAI 2023*, 2533–2542.

Yang, K., Singh, D., & Menczer, F. (2024). Characteristics and prevalence of fake social media profiles with ai-generated faces. *Journal of Online Trust and Safety*, *2*(4). https://doi.org/10.54501/jots.v2i4.197

Zhang, Y.-D., Wang, J., Wu, C.-J., Bao, M.-L., Li, H., Wang, X.-N., Tao, J., & Shi, H.-B. (2016). An imaging-based approach predicts clinical outcomes in prostate cancer through a novel support vector machine classification. *Oncotarget*, *7*(47), 78140–78151. https : / / doi . org / 10 . 18632 / oncotarget.11293

A Hyperparameter Grid Search

Table 7
$\text{CNN}_{\text{shallow}}$ Hyperparameter configurations and performance metrics.

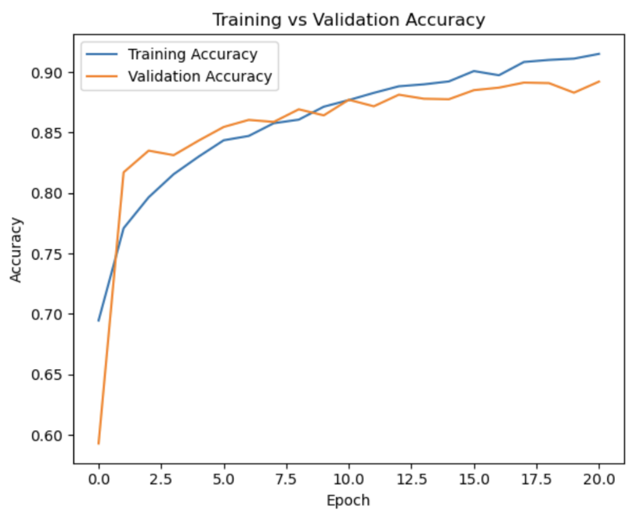
Kernel	Dropout	LR	Batch	Epochs	Train Acc	Val Acc	Train Loss	Val Loss
3x3	0.2	0.0001	32	15	0.9573	0.9142	0.1060	0.2125
5x5	0.2	0.0001	32	18	0.9575	0.9092	0.1043	0.2266
3x3	0.4	0.0001	32	19	0.9632	0.9071	0.0977	0.3334
5x5	0.4	0.0001	64	17	0.9345	0.9062	0.1546	0.2323
3x3	0.2	0.0001	64	18	0.9721	0.9033	0.0745	0.2613
3x3	0.4	0.0001	64	18	0.9468	0.8992	0.1366	0.2798
3x3	0.4	0.00001	32	25	0.9196	0.8908	0.1942	0.2460
5x5	0.2	0.00001	32	21	0.9342	0.8883	0.1657	0.2808
5x5	0.2	0.0001	64	16	0.9556	0.8879	0.1146	0.2988
3x3	0.2	0.00001	32	20	0.9357	0.8850	0.1661	0.2584
5x5	0.4	0.0001	32	20	0.9473	0.8846	0.1352	0.2728
5x5	0.4	0.00001	64	25	0.8969	0.8817	0.2373	0.2667
5x5	0.4	0.00001	32	20	0.9113	0.8808	0.2189	0.2718
3x3	0.4	0.00001	64	25	0.9146	0.8804	0.2153	0.2580
5x5	0.2	0.00001	64	21	0.9232	0.8767	0.2039	0.2803
3x3	0.2	0.00001	64	25	0.9226	0.8754	0.1916	0.2802

Table 8
CNN_{deep} Hyperparameter configurations and performance metrics.

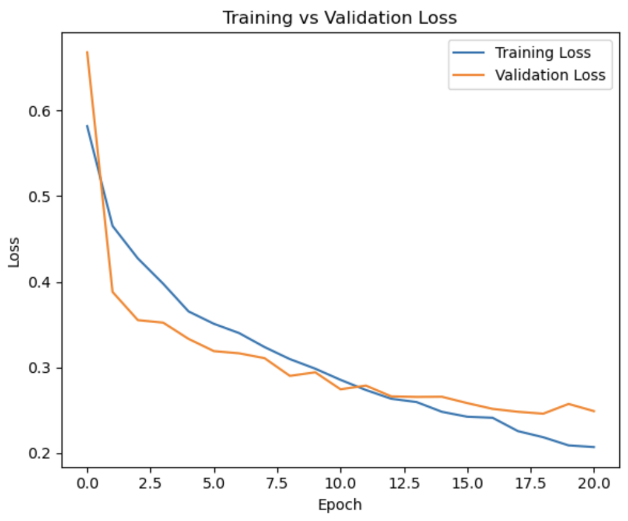
Kernel	Dropout	LR	Batch	Epochs	Train Acc	Val Acc	Train Loss	Val Loss
3x3	0.2	0.0001	64	19	0.987	0.947	0.042	0.151
3x3	0.2	0.0001	32	14	0.972	0.937	0.073	0.191
3x3	0.4	0.0001	32	16	0.977	0.936	0.063	0.191
5x5	0.2	0.0001	32	14	0.972	0.920	0.074	0.223
5x5	0.2	0.00001	32	25	0.950	0.918	0.133	0.208
3x3	0.2	0.00001	32	25	0.947	0.915	0.142	0.205
5x5	0.4	0.00001	32	25	0.924	0.908	0.187	0.226
3x3	0.4	0.0001	64	13	0.966	0.902	0.092	0.319
5x5	0.4	0.0001	64	15	0.912	0.900	0.216	0.296
3x3	0.4	0.00001	64	25	0.911	0.898	0.212	0.243
3x3	0.4	0.00001	32	25	0.920	0.892	0.190	0.250
5x5	0.2	0.00001	64	25	0.926	0.890	0.187	0.253
3x3	0.2	0.00001	64	25	0.915	0.866	0.211	0.291
5x5	0.2	0.0001	64	18	0.981	0.858	0.052	0.510
5x5	0.4	0.00001	64	25	0.864	0.858	0.320	0.316
5x5	0.4	0.0001	32	12	0.878	0.847	0.286	0.339

B Training performance of CNNs

CNN$_{\text{shallow}}$:



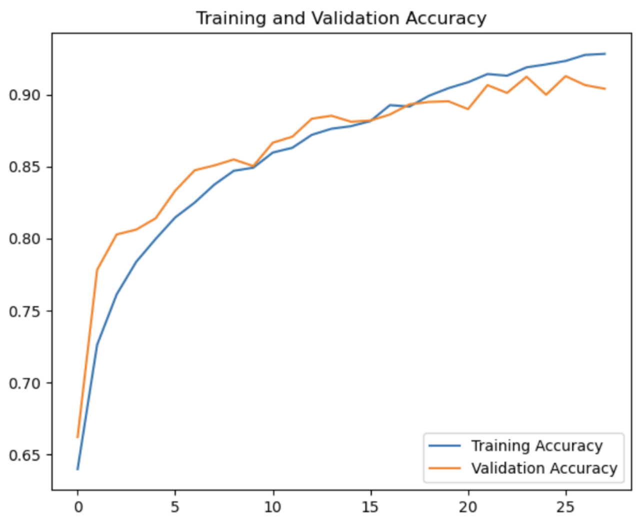
(a) Training vs. validation accuracy



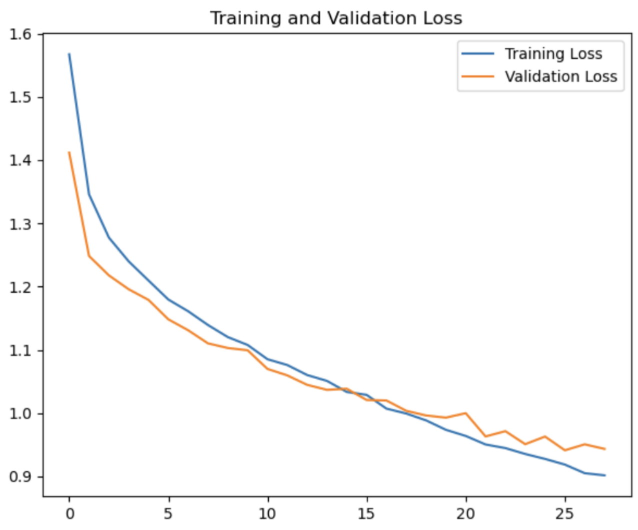
(b) Training vs. validation loss

Figure 5. Training performance of CNN$_{\text{shallow}}$. Accuracy and loss curves illustrate model learning progress across 21 epochs.

CNN$_{\text{deep}}$:



(a) Training vs. validation accuracy



(b) Training vs. validation loss

Figure 6. Training performance of CNN$_{\text{deep}}$. Accuracy and loss curves illustrate model learning progress across 28 epochs.