Electrical and Electronics Engineering
Semester Project
Master Semester 2 - Spring 2024

Exploring Diffusion-Generated Image Detection Methods

Student: Mehdi Abdallahi

Supervised by: Yuhang Lu
Prof. Dr. Touradj Ebrahimi

June 14, 2024

Multimedia Signal Processing Group
EPFL

Abstract

This research project investigates the current advancements in deep learning-based methods for detecting synthesized images. The research focuses on leveraging large pre-trained models to enhance the generalization of deepfake detection. In this study, alternative methods for enhancing the generalization capability of the detection will be explored and evaluated across multiple test sets.

The datasets used include entire face synthesis deepfakes generated by diffusion models and generative adversarial networks. Additional exploration is conducted using other types of fake images (identity swap, attribute manipulation, expression swap).

The repository containing the code used for this research is available here: `https://github.com/mehdi533/Deepfake-Detection.git`.

Contents

1 Objective of the Research

In recent years, face manipulation and generation techniques have achieved remarkable progress. The deep learning-based tools along with open-source software have simplified the creation of forgery, raising public concern about the potential abuse for malicious purposes. Computer vision and media security experts have devoted significant efforts to address the problem of face manipulation caused by deepfake techniques but the problem of detecting purely synthesized face images has been studied to a lesser extent. In particular, the recent popular diffusion models have shown remarkable success in image synthesis. Existing detectors struggle to detect images generated by diffusion models and suffer from the generalization problem.
The objective of this project is to first familiarize the student with the state-of-the-art deep learning-based synthesized image detection methods and databases. Then the student will explore using large pre-trained vision models to detect various synthesized images. It is also encouraged to explore other methods from different perspectives. At the end, the student should evaluate the proposed detection method on multiple test datasets to show both the performance and generalization ability.
In general, the following tasks shall be performed by the student:

1. Literature review of the state-of-the-art deep learning-based synthesized image detection methods and summarize their key ideas.

2. Set up a state-of-the-art synthesized image detection method as a baseline.

3. Explore using large pre-trained vision models to detect diffusion model-generated images and set up a training/validation/testing pipeline.

4. Assess the performance of the proposed detector on multiple test sets generated by different types of GANs and diffusion models.

5. Document the code and results and write a report on the project.

Deliverables (final report, final presentation, electronic package of all documents and software, etc.) must be submitted according to EPFL and MMSPG guidelines and schedule.

2 Introduction

The digital manipulation of images and videos has emerged as a significant public concern in recent years. Deepfake techniques present numerous potentially harmful applications, such as fake news, hoaxes, fake pornography, and financial fraud, highlighting the need for robust detection methods.
Deepfakes encompass four main types of facial manipulation techniques: entire face synthesis, attribute manipulation, identity swap, and expression swap. The existence of various methods to create fake images enhances the difficulty in creating a universal detector.

Figure 1: Images from the FaceForensics++ [21] dataset. Examples of: real image, Deepfake, Face2Face, FaceSwap, NeuralTextures, in order.

This project focuses on entire face synthesis deepfakes, but exploration of other types of deepfakes will also be conducted.
Entire face synthesis deepfakes can be created using different methods, with our focus being on Generative Adversarial Networks (GANs) and Diffusion Models (DMs):

- GANs consist of two neural networks: a discriminator that tries to differentiate real from fake images and a generator that tries to fool the discriminator, improving during training.

Figure 2: Images generated by GAN. Examples generated with: ProGAN [8], StyleGAN [7], VQGAN [3], in order.

- Diffusion models consist of a single neural network that progressively denoises a sample from pure noise until a realistic image is created.

Figure 3: Images generated by diffusion. Examples generated with: DDIM [24], DDPM [5], PNDM [11], LDM [20], in order.

This research focuses on large pre-trained convolutional neural networks (CNNs) and transformers. Additionally, different pre-processing techniques and transformations are studied, encompassing state-of-the-art detection methods. This study also plays a role in ensuring that people can trust digital media in this modern age where misinformation and fake images spread extremely rapidly.

3 Related Work

3.1 Robust detection of CNN-generated images with ResNet50

In a study on robust detection of synthesized images, Wang et al. published *CNN-generated images are surprisingly easy to spot... for now* [27]. The paper presented a universal detector to separate authentic images from those created by various CNN architectures, they trained a ResNet50 [4] pre-trained on ImageNet(v1) [22].

The method they proposed achieved a new benchmark for research on deepfake detection. These results suggest that a single classifier can effectively detect synthesized images coming from various CNN architectures, giving us motivation to train different architectures and observe how they generalize to different image generators.

3.2 Enhancing deepfake detection with orthogonal training of CNN ensembles

In *Detecting GAN-generated images by orthogonal training of multiple CNNs* published by Mandelli et al. [16] in 2022, GAN-generated image detection is addressed through an ensemble of CNNs. The focus is also on the generalization of synthesized image detectors to correctly classify images generated by unknown GANs. The method proposed uses multiple CNNs orthogonally trained, meaning that they comply with at least one of these conditions: the datasets include images depicting different semantic content (e.g., cats or humans); the datasets include images that underwent different post-processing (e.g., compression); the datasets include images that underwent different compressions (e.g., different JPEG implementations); the datasets include images synthesized by different GANs. This orthogonality helps in not overfitting to specific characteristics of the known generators, boosting the generalization capacity of the detector.

Detection is pursued by splitting the input image into patches and feeding them through the CNNs. For each patch, the CNN will generate a score indicating if its associated patch is real or fake. From the patch scores provided by a CNN for a single image, the highest score is recovered. Finally, all the CNNs' scores are averaged to have a final classification decision.

The paper further examines robust training techniques by applying strong data augmentations that mimic familiar image post-processing operations. This is crucial as it simulates real-world scenarios, where photos could be manipulated to hide synthetic traces. The experimental results demonstrate that such an approach effectively leads to high accuracy in detecting images created by the state-of-the-art GAN, StyleGAN3, even when these images are not associated with the training data.

Mandelli et al. emphasized the importance of orthogonality between any training datasets and the ensemble of CNNs to improve the robustness and the generalization capacity of detectors. This work achieved at the time the state-of-the-art of GAN detectors, setting a benchmark for future studies.

3.3 Enhancing generalization in fake image detection across generative models

In the research paper *Towards Universal Fake Image Detectors that Generalize Across Generative Models* [17] published by Ojha et al. in 2023, an unbiased generalizing approach to synthesized image detection is established. The study addresses the issue of the real class acting as a sink, including the fake images not originating from the same generator used to generate the training data.

To address this, Ojha et al. proposed a new method that uses a feature space defined by a large pre-trained vision transformer: CLIP ViT/14. After projecting the images on the feature space defined by the transformer, they use nearest neighbors and a linear classifier to do the final classification. Their approach significantly outperforms the previous state-of-the-art methods.

Their model presented significant improvements in generalization ability, resulting in better detection performance over a wide range of generative models. The paper strongly underlines that using a feature space, not fine-tuned to distinguish authentic and generated images, results in an effective backbone for universal fake image detection. During our study, different approaches will be used to train our vision models, visualizations of the differences in performance when using pre-trained feature spaces and training new ones will be presented.

3.4 Detection of AI-synthesized images

Lu and Ebrahimi published in 2024 *Towards the detection of AI-synthesized human face images* [14], in this research paper, they devised a benchmark for a comprehensive assessment of the ability of detectors to generalize against synthesized human faces at the state-of-the-art. The created benchmark consists of synthetic images obtained using two types of methods: GANs and diffusion models. They showed detectors trained from frequency representations to increase the performance across generative models significantly.

They also compared their methods to the ones published by Wang et al. in 2020 [27], Mandelli et al. in 2022 [16], and Ojha et al. in 2023 [17], which are methods that guided the research detailed in the next sections. The same datasets are used for the training and the assessment of the performances of the detectors used.

4 Proposed Method and Implementation

In our study, the main focus is to gather useful insights for the generalization of entire face synthesis detection. To address this task, a state-of-the-art synthesized image detection method was first established as a baseline, built on the architecture used by Wang et al's in *CNN-generated images are surprisingly easy to spot... for now* [27]. In order to train and assess the performances of a wide range of models, a convenient pipeline for training, validation and evaluation was also implemented. This allowed to move onto the next step: the exploration of large pre-trained vision models. More than 15 CNN architectures were explored, thanks to which interesting insights on the generalization of entire face synthesis classification were gathered, and using the best performing models, some augmentation techniques were explored to study their effect on the performance of the classification. At the end of the study, a model ensemble was developed to obtain the best performing detector possible.

The main ideas of the research done include:

- The investigation of several large pre-trained vision models, the goal being the extraction of meaningful insights, with a particular attention to the feature extraction hidden layers that come with large pre-trained vision models.
- The use of multiple models to vote together for the final classification.
- The use of augmentation techniques to understand their influence on the classification of entire face synthesis images.
- The generalization of these methods to other types of fake images and the exploration of the FaceForensics++ dataset [21].

4.1 Pipeline

The initial pipeline developed by Wang et al. [27] was designed for the training of a pre-trained ResNet50 model. For this implementation, part of the code served as the baseline, on which a pipeline was constructed to train a variety of models.

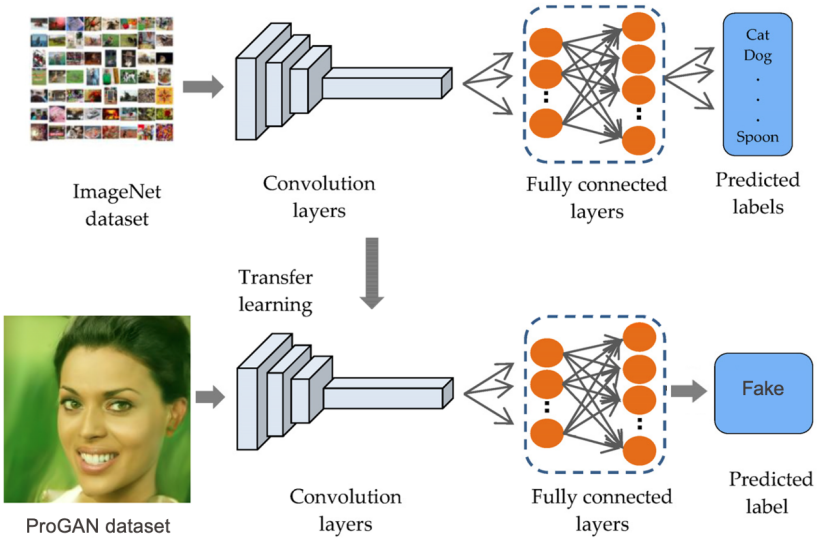

Figure 4: A visualization of the training pipeline used for the classification of the images. Models can be loaded pre-trained as shown here, or can be newly initialized. Original image at `https://www.mdpi.com/1424-8220/23/2/570`.

This pipeline allows for different types of training:

1. The pre-trained layers of the model can be frozen, utilizing the existing feature extraction space in the hidden layers. The only part trained is the classifier (fully connected layers), which connects the feature space to the output neuron.
2. The model can also be fine-tuned, allowing more flexibility.
3. Alternatively, the model can be trained from scratch.

This possibility of training models in different ways facilitated the study of the impact of pre-trained feature extraction, allowing the use of models for which the classification is less subject to overfitting on a particular GAN's fingerprint. The generalization performance for the different types of training can be well observed on tables 5 and 6 in the results section. Figure 5 presents the main stages of the pipeline in a comprehensive flowchart.

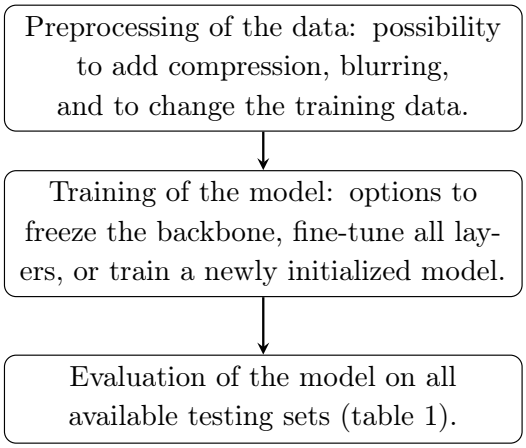

Figure 5: Image classification training and testing pipeline

At first, the idea was to train the models on two entire face synthesis datasets: DDIM [24] and ProGAN [8], but given the very high performances of several pre-trained models on the testing data (see table 2), it became difficult to qualitatively determine which one was performing better. For this reason, it was decided to train only on the ProGAN dataset, delving deeper in the challenge of generalization.

4.2 Exploring large pre-trained vision models

4.2.1 VGG16

VGG16 [23] is a classic CNN, known for its simplicity and depth, making it a good candidate for generalizing across different entire face synthesis generation methods. The main advantage when it was proposed in 2015 was that it leverages a very deep network with small convolution filters, allowing it to learn intricate details while maintaining computational efficiency.

The weights used for the freezed backbone and fine-tuning trainings are pre-trained on the ImageNet-1K(v1) dataset [22], the model is loaded using the torchvision library [15].

4.2.2 ResNet50

ResNet50 [4] was the first model trained and tested, it was the one implemented in the baseline pipeline by Wang et al. [27], and it provides a strong baseline for image classification tasks. ResNets can be scaled to balance computational efficiency and model performance (e.g. ResNet18

or ResNet101). ResNets introduced residual learning blocks in 2016, which improve information flow and training efficiency, giving them a good ability to capture complex features and thus accurately classify synthesized images.

The weights used for the freezed backbone and fine-tuning trainings are pre-trained on the ImageNet-1K(v2) dataset [19], the model is loaded using the torchvision library.

4.2.3 ResNeXt

ResNeXt [31] builds on the ResNet architecture, but has the particularity to learn more features through cardinality. In 2017, ResNeXt introduced this concept, referring to the number of transformations in each layers, allowing the model to capture diverse features. This model achieved better performances than ResNet with fewer parameters, making it a more efficient model for complex tasks. The motivation to use this model is to observe how capturing complex patterns influences the generalization of the detector.

The model is pre-trained on the ImageNet-1K(v1) dataset and is loaded from the HuggingFace hub, using the timm library [28], the model name is `timm/resnext101_32x8d.tv_in1k`.

4.2.4 EfficientNet

The motivation to try EfficientNet models [25] comes from their balance between performance and computational efficiency, it is achieved through an efficient architecture design and compound scaling. The version tested are EfficientNet-B0 and EfficientNet-B4, the difference between them being that EfficientNet-B0 is used for limited computational resources, whereas EfficientNet-B4 is adapted for situations where computational resources are less of a constraint and performance is of greater importance. EfficientNet models introduced in 2019 a new method to uniformly scale all dimensions. At the time, it achieved state-of-the-art performances while using fewer parameters and having a faster inference time compared to previous models.

The models are pre-trained on the ImageNet-1K(v1) dataset and both of them are loaded using the torchvision library.

4.2.5 RegNet

RegNet models [18] are designed to provide high efficiency and performance, they were introduced in 2020 and are powerful and adaptable models. The architecture of a RegNet is characterized by its use of simple, regular blocks that can be easily scaled to adjust for various computational budgets.

The models are pre-trained on the ImageNet-1K(v2) dataset [19] and the `regnet_y_400mf` version is used and loaded using the torchvision library.

4.2.6 Big Transfer

Big Transfer (BiT) [9] is a large-scale pre-trained model designed to facilitate transfer learning, allowing it to adapt well to new tasks with minimal re-training. This model was proposed in 2020 and stood out due to its great performances on downstream tasks. This is very interesting for our task as the objective is the generalization of the detection of generated images.

The model is pre-trained on the ImageNet-1K(v1) dataset and is loaded from HuggingFace's transformers library [29], the base version is used here, with `google/bit-50`.

4.2.7 ViT

In 2020, Vision Transformer (ViT) [30] proposed a new application of transformer architectures for image recognition tasks. The images are divided in patches and are processed with transformer encoders, similarly to token embeddings in natural language processing transformers. This new approach demonstrated state-of-the art performance by transformers in vision tasks.

The models are pre-trained on the ImageNet-1K(v1) dataset and they are loaded from HuggingFace's transformers library, four sizes are proposed (tiny, small, base, large) but only base version is used here, with `google/vit-base-patch16-224`.

4.2.8 DeiT

DeiT (Data-efficient Image Transformer) [26] is a model introduced in 2021 and distinguishes itself by its ability to train vision transformers efficiently, without requiring important computational resources usually needed for ViT. DeiT utilizes distillation techniques to enhance training efficiency and performance. The interest for us is to see if the performance of DeiT will match the performance of ViT while using less computational resources.

The model used is pre-trained on the ImageNet-1K(v1) dataset and loaded from HuggingFace's transformers library, four sizes are proposed (tiny, small, base, large) but only the base version is used here, with `facebook/deit-base-distilled-patch16-224`.

4.2.9 BeiT

BeiT (Bidirectional Encoder representation from Image Transformers) [1] was introduced in 2021 and proposed an innovative self-supervised pre-training method by treating image patches as discrete visual tokens and learning their representations through a masked image modelling, similar to what is done in NLP tasks. This method of using generalizable image representations makes of BeiT a good candidate for our exploration.

The model is pre-trained on the ImageNet-1K(v1) dataset and loaded from HuggingFace's transformers library, the base version is used here, with `microsoft/beit-base-patch16-224`.

4.2.10 CoAtNet

CoAtNet [2] was introduced in 2021 and provides a hybrid architecture, that combines convolution and attention mechanisms, allowing to capture both local and global features to correctly classify different type of images. CoAtNet utilizes the strengths of both convolutions and transformers and provides a very robust extraction of diverse features.

The model is pre-trained on the ImageNet-1K(v1) dataset and is loaded from the HuggingFace hub, using the timm library, the model name is `coatnet_0_rw_224.sw_in1k`.

4.2.11 Swin Transformers

Shifted Window Transformers [12] are a particular type of transformers that were introduced in 2021 and use the shifted window method. The window-based self-attention contrasts with the global self-attention and reduces the computational complexity, shifted windows allow for cross-window connections and a better capture of spatial relationships. The innovative architecture of Swin transformers' allows for a flexible classification, ideal for generalization. There are different sizes of Swin transformers that can be adapted to the computational resources available, in this case the *tiny* transformer was prioritized, the *large* Swin transformer was also tested but required more consequent computational resources for training.

The models are pre-trained on the ImageNet-1K(v1) dataset and they are loaded from HuggingFace's transformers library: `microsoft/swin-tiny-patch4-window7-224` for *Swin tiny*, `microsoft/swin-base-patch4-window7-224` for *Swin base*, and `microsoft/swinv2-large-patch4-window12to16-192to256-22kto1k-ft` for *Swin large*.

4.2.12 ConvNeXt

ConvNeXt [13] combines the power of CNNs and modern architectures inspired by transformers. It was proposed in 2022, it is the most recent architecture explored in the following sections and its ability to integrate the best parts of CNNs and vision transformers allowed it to achieve state-of-the-art performances.

The models are pre-trained on the ImageNet-1K(v1) dataset, four sizes are proposed (tiny, small, base, large), and the torchvision library is used to load them.

4.3 Classifier and initialization of the weights

The classifier trained when using a frozen backbone contains two fully connected layers that are defined as follows:

1. The first layer of the classifier takes the flattened output from the convolutional layers as input. With x representing the flattened feature vector from the convolutional layers, with a dimension of d, the first fully connected layer transforms x into a 64-dimensional vector h, it then passes through a ReLU activation.

2. The second layer of the classifier takes the 64-dimensional vector h from the first fully connected layer and reduces it to a single output y that is the result of: $y = W_2h + b_2$, with W_1 and b_2 the weight matrix and the bias scalar of the second layer.

When the backbone is not frozen, only one fully connected layer connects the flattened output from the layers to the output. When then weights are newly initialized, a normal initialization is used, with mean 0 and standard deviation set at 0.2.

4.4 Augmentation techniques

In this study, two types of augmentations were studied and implemented, the objective being to improve the performances of the classification to the maximum attainable. In several research papers, notably in Wang et al's [27], it is shown that blurring and compressing part of the dataset during training can lead to better performing models. The two possible augmentation explored here are the following:

- JPEG compression, it is applied the using opencv library [6], when loading the data, the proportion that will be compressed is passed as argument (default is 0%). The compression is applied before normalization and the quality is of 75 for all compressed images. A visualization of the effect of compressing images is shown below on figure 6.

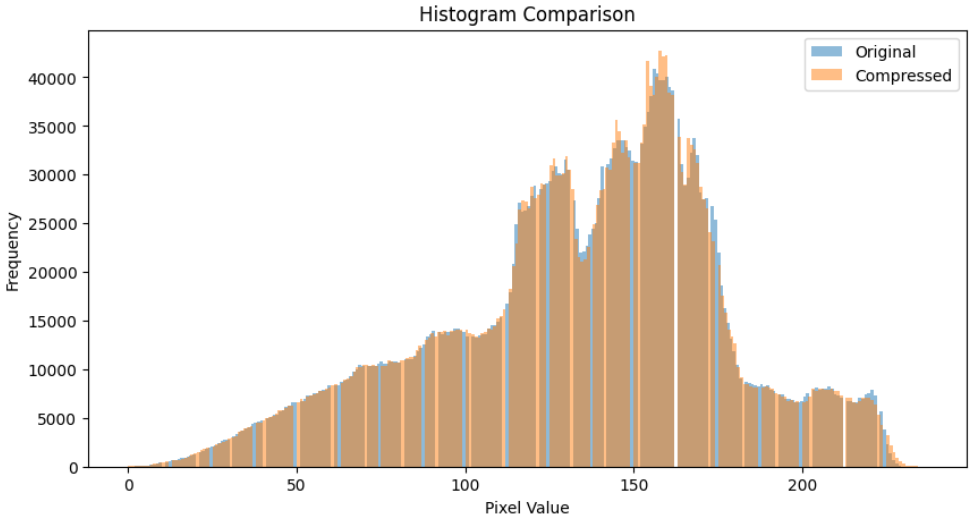

Figure 6: Histogram that shows the absence of certain values of pixels when compressing a synthesized image from StyleGAN2.

- Blurring is also applied using opencv library [6], when loading the data, the proportion that will be blurred is passed as argument (default is 0%). The blurring is a Gaussian blurring applied with a kernel of (5,5) for all images. A visualization of the blurring effect is shown below on figure 7.

Figure 7: Comparison between original (left) and blurred image (right) from the LDM dataset.

4.5 Model Ensembles

The objective of the *Model Ensembles*, is to use different models to complement each other, using them to predict the class of an image. This technique is inspired by the work done by Mandelli et al. [16], although their approach uses orthogonally trained models, implying the use of different datasets, our implementation uses different models, all trained on the same dataset.

The principle is the following: when predicting the class of the image, the models predictions are used to decide for the final prediction, this process is refered to as the *voting* of the models. This can be done by *simple averaging*: $P_{\text{final}} = \frac{1}{n}\sum_{i=1}^{n} P_i$, or by *weighted averaging*: $P_{\text{final}} = \sum_{i=1}^{n} w_i \cdot P_i$, with $\sum_{i=1}^{n} w_i = 1$.

5 Results

In this section, the details of the implementation will be given, after which the results for the different training methods mentioned will be presented and commented. The results on the GAN and diffusion-generated images will then be provided and finally, the results of the exploration on the FaceForensics++ dataset will be presented.

5.1 Implementation Details

5.1.1 Dataset

Table 1 presents the different datasets used for this study. The main datasets used for training are ProGAN [8], DDIM [24], and the ones from the FaceForensics++ dataset. Other trainings were made using PNDM [11], DDPM [5] and LDM [20] during early tests. The generalization of a model can be estimated by observing the performance of the classification on other types of image generators. For instance, when training on ProGAN, the generalization capability of the trained model is showcased by its testing performance on diffusion generated images. The FaceForensics++ dataset [21] contains other types of fake images; Face2Face, FaceSwap, and NeuralTextures are deepfakes that are obtained using other methods which makes it very challenging to correctly classify these images when training on entire image synthesis images.

Family	Dataset	Class	train/val/test	Size	Format
GANs [14]	ProGAN [8]	fake	38k / 1k / 1k	256x256	PNG
	StyleGAN2 [7]	fake	38k / 1k / 1k	256x256	PNG
	VQGAN [3]	fake	38k / 1k / 1k	256x256	PNG
Diffusion Models [14]	DDIM [24]	fake	38k / 1k / 1k	256x256	PNG
	PNDM [11]	fake	38k / 1k / 1k	256x256	PNG
	DDPM [5]	fake	38k / 1k / 1k	256x256	PNG
	LDM [20]	fake	38k / 1k / 1k	256x256	PNG
Celebrity HQ [10]	CelebA-HQ-img	real	28k / 1k / 1k	1024x1024	JPEG
FaceForensics++ [21]	Original	real	64k / 1.4k / 1.4k	256x256	PNG
	Deepfakes	fake	64k / 1.4k / 1.4k	256x256	PNG
	Face2Face	fake	64k / 1.4k / 1.4k	256x256	PNG
	FaceSwap	fake	64k / 1.4k / 1.4k	256x256	PNG
	NeuralTextures	fake	64k / 1.4k / 1.4k	256x256	PNG

Table 1: Datasets used in this study

5.1.2 Hyperparameters

The optimizer used for training is Adam, as it presented better results than SGD during early tests. The parameter β_1 that controls the decay rate for the moving average of the gradient is set at 0.9 and the parameter β_2 that controls the moving average of the squared gradient is set at 0.999. A learning rate scheduler is used, the learning rate is initially set at 10^{-4} and is lowered by a factor 10 if the average precision (AP) on the validation set doesn't change by more than 0.1% during 5 epochs. After this first change, it will change again to 10^{-6} if the AP on the validation set doesn't change by more than 0.2% during 5 epochs. If the same condition is filled again during 5 epochs, the training stops. The loss function used is BCEWithLogitsLoss. The batch size used for training is 256 for ResNet50, VGG16, EfficientNetb0, and Swin Tiny; it is 128 for EfficientNetb4, ViT Base, DeiT Base, BeiT Base, ResNeXt, ConvNeXt, and RegNet 400mf; and 32 for CoAtNet, if the batch size is exceptionally different to the one given here, it is mentioned in the caption.

5.1.3 Training Data

During early tests, models were fine-tuned on images generated by a diffusion model (DDIM) and by a GAN (ProGAN), leading to very good results on several testing sets, making it difficult to determine which model was performing better between EfficientNet b4, Swin Transformers and BiT. For this reason, it was decided to change the training data to a set generated by a single GAN (ProGAN), making the generalization more challenging and thus, showcasing the generalization ability of each model better. Table 2 presents the AP for models implemented early in the study and additional models were explored later on.

Model	StyleGAN	VQGAN	PNDM	LDM	DDPM	Average
ResNet50	78.49	100.00	100.00	85.93	100.00	92.28
VGG16	44.71	100.00	99.99	85.01	100.00	85.94
EfficientNet b0	72.11	100.00	99.99	53.87	100.00	85.99
EfficientNet b4	98.71	100.00	99.99	97.13	100.00	99.57
Swin Tiny	**99.91**	100.00	100.00	99.87	100.00	**99.96**
Swin Base	99.30	100.00	100.00	**99.99**	100.00	99.86
Swin Large	40.44	100.00	100.00	99.96	100.00	88.88
BiT	96.00	100.00	99.99	98.79	100.00	98.96
ResNext	50.04	100.00	99.99	72.66	100.00	84.54
CoAtNet	47.86	100.00	100.00	42.83	100.00	78.14

Table 2: AP (in %) for different fine-tuned models (training data: DDIM, ProGAN, Celeb-HQ), tested on images generated by GANs and DMs.

The Swin Base and Swin Large models were not followed up with as the trade off between computational costs and performance appeared to be much better for Swin Tiny.

5.2 Comparison of training methods

5.2.1 The different training modes (fine-tuning, frozen backbone, from scratch)

To effectively train a model and achieve the best generalization possible, several training methods were studied. The pre-trained aspect of a neural network is an important feature to ensure better generalization for the classification of data that doesn't come from the same distribution, a qualitative and quantitative dataset for pre-training is also essential. The motivation of this section is to determine the training mode that should be prioritized for future explorations.

The following boxplots showcase the average precision achieved by all the models used on different datasets during testing (only relevant plots are shown here). The models have no difficulty achieving almost perfect performances on ProGAN (used for training) and VQGAN. Figure 8 presents the average precision on the StyleGAN testset; the first observation is that models trained with newly initialized weights achieve very low scores, whereas the scores obtained by fine-tuned models and models trained with a frozen backbone are much higher. Another observation is that fine-tuned models present more variance, but perform slightly better on average. The score for each model tested on GAN generated images is available in table 5.

When testing on diffusion models, the results are more noisy. It is not as clear as it is when testing on StyleGAN but it still shows that some fine-tuned models are able to achieve a perfect average precision score, making it our preferred training method for future exploration. The boxplots on figures 9, 10, 11, and 12 present the average precision achieved by different models on diffusion generated testing sets. The score for each model tested on diffusion generated images is available in table 6.

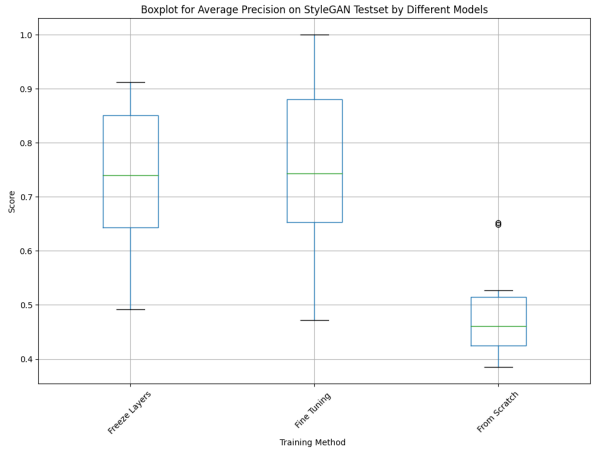

Figure 8: Boxplot for AP (y-axis) on StyleGAN testset obtained by different models for different training modes (x-axis) (training data: ProGAN, Celeb-HQ)

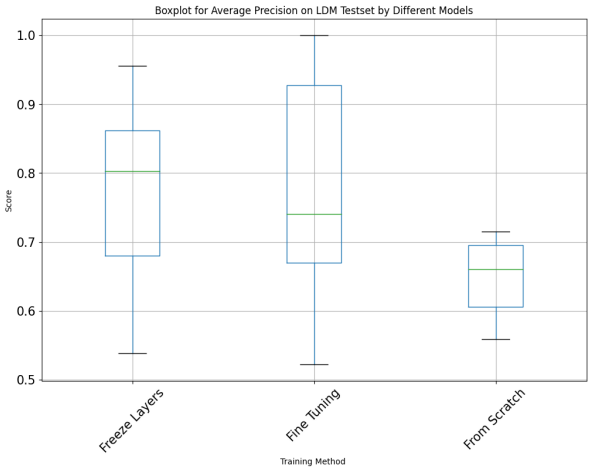

Figure 9: Boxplot for AP (y-axis) on LDM testset obtained by different models for different training modes (x-axis) (training data: ProGAN, Celeb-HQ)

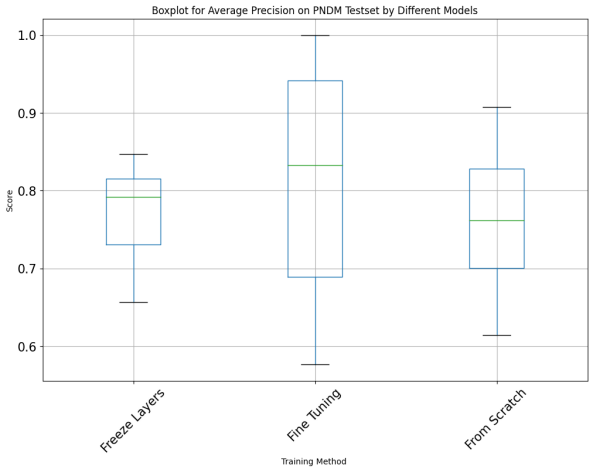

Figure 10: Boxplot for AP (y-axis) on PNDM testset obtained by different models for different training modes (x-axis) (training data: ProGAN, Celeb-HQ)

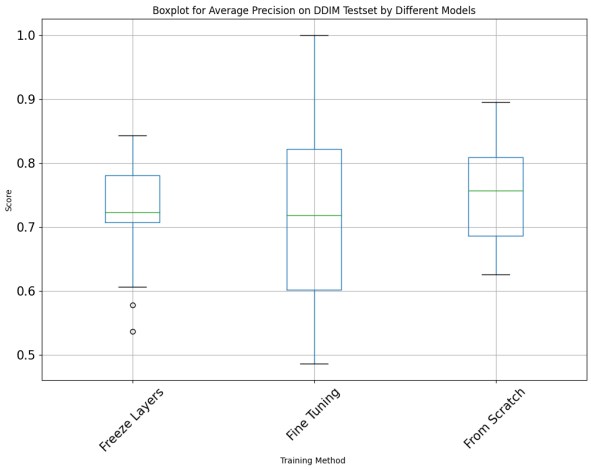

Figure 11: Boxplot for AP (y-axis) on DDIM testset obtained by different models for different training modes (x-axis) (training data: ProGAN, Celeb-HQ)

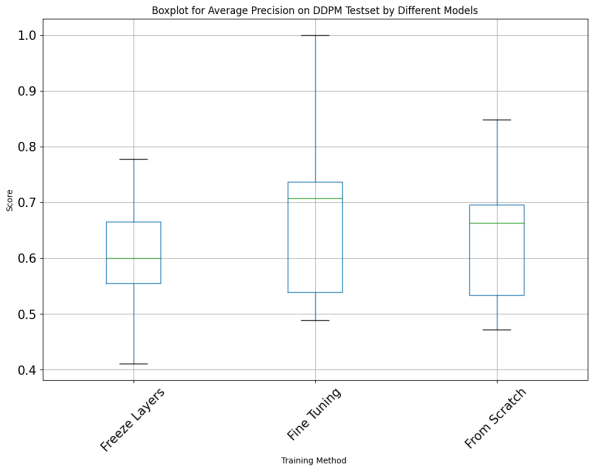

Figure 12: Boxplot for AP (y-axis) on DDPM testset obtained by different models for different training modes (x-axis) (training data: ProGAN, Celeb-HQ)

5.2.2 Pre-processing techniques

The augmentation techniques used are detailed in the previous sections. Based on the results observed in table 3, compressing the images for training immediately impacts heavily the performance of the model when classifying images from the StyleGAN2 test set. Blurring on the other hand seems to positively impact the classification of images from StyleGAN2 but performance drops when blurring 60% or more of the training data.

Percentage transformed		**Testing Dataset**			
Compression	**Blurring**	**ProGAN**	**StyleGAN**	**VQGAN**	**Average**
-	-	100.00	96.37	100.00	98.79
20%	-	100.00	51.41	100.00	83.80
40%	-	100.00	55.97	100.00	85.32
60%	-	100.00	46.36	100.00	82.12
80%	-	100.00	51.63	100.00	83.88
100%	-	100.00	47.39	100.00	82.46
10%	10%	100.00	68.51	100.00	89.50
-	**20%**	**100.00**	**98.97**	**100.00**	**99.66**
-	**40%**	**100.00**	**98.60**	**100.00**	**99.53**
-	60%	100.00	89.24	100.00	96.41
-	80%	100.00	94.35	100.00	98.12
-	100%	100.00	92.41	100.00	97.47

Table 3: AP (in %) for Swin Tiny (batch size 128, fine-tuned on ProGAN and CelebA-HQ-img), tested on ProGAN, StyleGAN, and VQGAN datasets.

The scores obtained when testing on diffusion models are displayed in table 4. The insights obtained from these tests are that compressing images moderately (around 40%) seems to improve the classification, with the exception of the LDM test set where the performance is lower when compressing the training data. Blurring also moderately appears to improve the classification on diffusion models.

Percentage transformed		**Testing Dataset**				
Compression	**Blurring**	**DDIM**	**DDPM**	**PNDM**	**LDM**	**Average**
-	-	71.92	69.31	89.57	96.50	81.33
20%	-	81.84	78.46	84.40	84.13	82.71
40%	-	86.87	86.77	87.15	78.02	84.20
60%	-	83.13	79.75	85.46	70.84	79.80
80%	-	82.49	81.19	83.90	71.63	79.80
100%	-	71.59	67.99	70.73	72.97	70.82
10%	10%	89.31	82.44	91.22	79.52	85.62
-	**20%**	**90.27**	**84.35**	**98.02**	**99.16**	**92.95**
-	**40%**	**90.94**	**79.12**	**97.27**	**99.05**	**91.10**
-	60%	80.67	76.43	94.19	94.27	86.39
-	80%	59.82	50.74	81.11	93.74	71.35
-	100%	51.73	49.84	68.77	93.28	65.91

Table 4: AP (in %) for Swin Tiny (batch size 128, fine-tuned on ProGAN and CelebA-HQ-img), tested on DDIM, DDPM, PNDM, and LDM datasets

These findings suggest that blurring could improve the overall performance of our detector.

5.3 Results on entire face synthesis (GANs and diffusion models)

In this part, the results of the large pre-trained vision models are reported and the performances are assessed on multiple test sets generated by different type of GANs and diffusion models. The results presented correspond to the models detailed in the previous sections, all of them are trained using the images generated by the ProGAN [8] and CelebA-HQ-img [10] datasets. The average precision is presented for each mode of training, table 5 presents the results obtained on test sets generated by different types of GANs, table 6 presents the results obtained on test sets generated by different diffusion models.

5.3.1 General observations

The results obtained on GAN generated images (table 5) show that models trained on ProGAN [8] are challenged the most by the StyleGAN2 dataset [7], although this dataset contains images generated using the same type of generator (GAN) as the one in the training set. On VQGAN [3], all models distinctly separate real from generated images.

The results in table 6 show how well each model generalizes to diffusion generated images and in this table it is easier to spot models that have good generalizing potential. For instance, the fine-tuned Swin tiny, ConvNeXt, and VGG16 models achieve very good performances on those diffusion generated test sets. It appears that the models trained using ProGAN present greater difficulty on the test sets generated by StyleGAN2, DDIM, and DDPM.

5.3.2 Insights on the performance of the models

ResNet50 is the first model that was implemented, with the idea being to use it as a baseline and compare it to the other models performances. Although ProGAN and VQGAN are two image generators that present different architectures, ResNet50 presented very good performances on VQGAN's test set across all training strategies, the same goes for all the other models. Its performance on StyleGAN2 was lower but remained pretty high when training only the fully connected layers (frozen backbone) and fine-tuning. When testing on diffusion-generated images, ResNet50's performance was significantly lower, showing its low generalization capability.

As expected due to the depth of the network, VGG16 was indeed able to generalize a lot better than ResNet50 on all datasets. This time, the fine-tuning training method presented significantly better scores than the two other methods, highlighting the importance of pre-trained feature extractions and flexibility in training.

Given the different objectives of optimization of the EfficientNets, it is of no surprise that EfficientNet-B4 performed better than EfficientNet-B0 on the StyleGAN2 test set and on the images generated by diffusion models. EfficientNet-B4 was the first model tested that had an almost perfect score on images generated by StyleGAN2 and although the performance on the DDIM and DDPM sets is moderately high, it was higher for the PNDM and LDM sets when fine-tuning the model. Based on those few observations alone, it was already noticeable that there is a tendency of generalizing better when fine-tuning large pre-trained vision models compared to the other types of training.

A similar model to EfficientNet-B0 that was tested is the RegNet 400MF (RegNets also exist in different sizes depending on the resources available). The RegNet 400MF model performed slightly better than EfficientNet-B0, which would suggest that the bigger RegNet models (e.g. 4GF or 8GF) that are more similar to Efficient-B4 could yield better results but EfficientNet-B4 was outperformed by more recent models.

Model		GAN Dataset			Average
		ProGAN	StyleGAN	VQGAN	
ResNet50	Frozen Backbone	99.79	90.08	99.79	96.55
	Fine Tuning	100.00	81.88	100.00	93.96
	From Scratch	99.97	46.08	99.97	81.34
ResNext	Frozen Backbone	99.47	73.95	99.47	90.96
	Fine Tuning	100.00	60.26	100.00	86.75
	From Scratch	99.65	51.42	99.65	83.57
VGG16	Frozen Backbone	91.21	49.16	91.21	77.86
	Fine Tuning	100.00	60.08	100.00	86.69
	From Scratch	99.98	65.20	99.98	88.72
RegNet	Frozen Backbone	97.70	72.02	97.70	89.81
	Fine Tuning	100.00	85.19	100.00	95.06
	From Scratch	99.83	47.16	99.83	82.60
EfficientNet b0	Frozen Backbone	98.37	61.29	98.37	86.68
	Fine Tuning	100.00	74.34	100.00	91.45
	From Scratch	99.99	52.63	99.99	84.87
EfficientNet b4	Frozen Backbone	98.16	78.88	98.16	91.73
	Fine Tuning	100.00	98.86	100.00	**99.62**
	From Scratch	99.95	49.81	99.95	83.91
ViT	Frozen Backbone	98.27	62.41	98.27	86.32
	Fine Tuning	100.00	74.10	100.00	91.37
	From Scratch	100.00	39.22	100.00	79.74
DeiT	Frozen Backbone	99.80	73.82	99.80	91.81
	Fine Tuning	100.00	72.94	100.00	90.98
	From Scratch	99.99	38.51	99.99	79.50
BeiT	Frozen Backbone	90.87	64.30	90.87	82.01
	Fine Tuning	100.00	47.14	100.00	82.38
	From Scratch	100.00	45.31	100.00	81.77
CoAtNet	Frozen Backbone	99.19	85.11	99.19	94.50
	Fine Tuning	100.00	65.35	100.00	88.45
	From Scratch	100.00	64.91	100.00	88.30
BiT	Frozen Backbone	98.28	76.23	98.28	90.93
	Fine Tuning	100.00	**99.39**	100.00	**99.80**
	From Scratch	100.00	42.45	100.00	80.82
Swin tiny	Frozen Backbone	99.66	85.13	99.66	94.82
	Fine Tuning[1]	100.00	**99.99**	100.00	**99.99**
	From Scratch	100.00	45.05	100.00	81.68
ConvNext	Frozen Backbone	99.92	91.27	99.92	97.04
	Fine Tuning	100.00	88.07	100.00	96.02
	From Scratch	100.00	40.17	100.00	80.06

Table 5: AP (in %) for different models and training methods, tested on ProGAN, StyleGAN, and VQGAN datasets (training set: ProGAN and CelebA-HQ-img).

[1] For this training, the batch size used is of 256, when testing for the best pre-processing techniques, the batch size was of 128, explaining the difference between the results shown here and in the previous table.

Model		Diffusion Models Dataset				Average
		DDIM	DDPM	PNDM	LDM	
ResNet50	Frozen Backbone	57.79	54.48	66.86	82.93	65.52
	Fine Tuning	62.28	53.89	68.90	84.73	67.45
	From Scratch	76.66	66.30	76.21	66.82	71.50
ResNext	Frozen Backbone	70.69	60.01	73.08	73.65	69.36
	Fine Tuning	49.09	51.51	57.67	58.73	54.25
	From Scratch	87.20	78.52	86.60	68.30	80.16
VGG16	Frozen Backbone	83.42	66.50	79.19	75.79	76.23
	Fine Tuning	94.08	88.70	94.08	87.77	91.16
	From Scratch	86.05	75.81	86.41	69.72	79.50
RegNet	Frozen Backbone	71.56	55.44	83.28	65.31	68.90
	Fine Tuning	48.61	48.87	69.39	56.19	55.76
	From Scratch	80.87	68.83	82.82	66.06	74.65
EfficientNet b0	Frozen Backbone	72.27	57.48	78.40	53.84	65.50
	Fine Tuning	49.20	52.45	60.33	52.23	53.55
	From Scratch	63.27	56.43	61.94	63.57	61.30
EfficientNet b4	Frozen Backbone	60.66	51.77	69.33	67.98	62.44
	Fine Tuning	71.82	61.14	94.20	92.74	80.00
	From Scratch	75.69	68.68	75.38	60.59	70.09
ViT	Frozen Backbone	84.29	77.83	81.58	83.14	81.71
	Fine Tuning	77.52	72.34	80.36	74.00	76.06
	From Scratch	74.92	63.42	78.31	62.79	69.86
DeiT	Frozen Backbone	73.45	57.19	78.28	89.57	74.62
	Fine Tuning	60.17	56.73	63.46	74.10	63.61
	From Scratch	66.40	53.33	68.40	55.86	61.00
BeiT	Frozen Backbone	79.95	71.97	81.30	64.13	74.34
	Fine Tuning	79.44	73.70	83.25	69.86	76.56
	From Scratch	76.48	69.58	80.72	71.47	74.56
CoAtNet	Frozen Backbone	78.10	66.67	79.90	80.32	76.25
	Fine Tuning	82.21	70.76	86.68	66.97	76.66
	From Scratch	89.53	84.85	90.75	70.62	84.19
BiT	Frozen Backbone	74.70	61.24	84.75	86.15	76.71
	Fine Tuning	69.31	71.39	96.27	98.71	83.92
	From Scratch	62.58	47.14	61.44	59.30	57.62
Swin Tiny	Frozen Backbone	53.66	41.08	65.68	95.56	64.00
	Fine Tuning[1]	90.21	91.29	**99.85**	**99.88**	95.31
	From Scratch	69.94	51.58	72.71	57.17	62.85
ConvNext	Frozen Backbone	72.21	65.71	84.59	95.52	79.51
	Fine Tuning	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
	From Scratch	68.58	53.33	70.05	69.57	65.38

Table 6: AP (in %) for different models and training methods, tested on DDIM, DDPM, PNDM, and LDM datasets (training set: ProGAN and CelebA-HQ-img).

[1] For this training, the batch size used is of 256, when testing for the best pre-processing techniques, the batch size was of 128, explaining the difference between the results shown here and in the previous table.

The Swin Transformers presented the best results over the images generated by GANs; they have the highest score obtained on images generated by StyleGAN2. The generalization over diffusion models was also very good but left room for improvement on the DDIM and DDPM datasets. As shown in the augmentation section, some pre-processing techniques can help achieve better results on the DDIM and DDPM dataset for this transformer, making it the second best on the list for generalization over diffusion-generated images.

Big Transfer performed very accurately on the GAN-generated test sets when fine-tuned on ProGAN but surprisingly didn't perform so well when generalizing to diffusion-generated images. Using pre-processing techniques tested and presented in the previous sections could be a way to improve the results for this model on DDIM and DDPM and to make the best off of it's generalization potential.

The Vision Transformers (ViT, DeiT, BeiT) did not present very impressive generalization capability. When tested on diffusion-generated images, ViT and BeiT presented better results than DeiT, probably because this model is more efficiency oriented than the others. Overall, ViT still presented solid results, but several models outperformed it.

CoAtNet, quite surprisingly, is one of the models (along with ResNeXt) that reached the highest performance on diffusion-generated images when trained with ProGAN form scratch. The low batch size (32) used when training CoAtNet might play a role in the performance obtained when testing with this model.

Finally, ConvNeXt, the latest model combining CNNs and inspired by the architecture of the Swin transformers presented impressive generalization on diffusion-generated images, it scored perfectly on all the diffusion test sets in the dataset. This model obtained a performance slightly lower than Swin tiny on the StyleGAN2 test set, for this type of case, an exploration of Model Ensembles inspired by the paper *Detecting Gan-Generated Images by Orthogonal Training of Multiple CNNs* [16] could be interesting to make the best out of the predictions of the best performing models explored.

5.4 Exploration of the FaceForensics++ dataset

Table 7 presents the results obtained by the same models and training methods shown before, but tested on the FaceForensics++ datasets.

Model		**FaceForensics++ Dataset**				**Average**
		Deepfakes	Face2Face	FaceSwap	NeuralText	
ResNet50	Frozen Backbone	52.09	50.84	54.10	52.46	52.87
	Fine Tuning	58.24	53.46	50.26	55.59	54.89
	From Scratch	53.79	54.62	48.94	55.74	53.27
ResNext	Frozen Backbone	54.33	54.23	54.90	57.03	55.12
	Fine Tuning	49.54	54.81	50.88	57.14	53.09
	From Scratch	**60.74**	52.17	51.95	53.62	54.62
VGG16	Frozen Backbone	49.91	49.11	52.18	48.98	50.05
	Fine Tuning	54.69	58.44	51.16	57.86	55.04
	From Scratch	58.48	54.22	46.23	56.90	53.46
RegNet	Frozen Backbone	50.23	49.42	55.67	51.18	51.12
	Fine Tuning	51.93	56.70	55.66	61.30	56.40
	From Scratch	54.75	50.15	51.20	51.07	51.79
EfficientNet b0	Frozen Backbone	51.32	52.39	50.30	52.37	51.60
	Fine Tuning	52.73	54.57	60.21	58.42	56.98
	From Scratch	50.11	50.21	46.91	50.80	49.51
EfficientNet b4	Frozen Backbone	51.63	54.00	52.09	51.67	52.35
	Fine Tuning	50.21	55.37	**61.20**	56.30	55.27
	From Scratch	53.61	51.26	49.77	53.39	51.51
ViT	Frozen Backbone	56.75	55.06	54.37	57.57	55.94
	Fine Tuning	51.12	53.43	52.74	57.19	53.12
	From Scratch	48.51	49.60	47.92	48.73	48.69
DeiT	Frozen Backbone	52.06	52.94	53.39	56.24	53.66
	Fine Tuning	52.83	50.29	49.21	54.65	51.75
	From Scratch	50.00	48.74	49.77	49.78	49.57
BeiT	Frozen Backbone	49.53	52.69	55.10	51.18	52.13
	Fine Tuning	49.23	53.00	48.87	56.51	51.90
	From Scratch	51.23	51.95	50.19	55.55	52.23
CoAtNet	Frozen Backbone	55.15	53.47	51.49	53.51	53.40
	Fine Tuning	51.40	52.46	54.78	55.63	53.07
	From Scratch	51.05	53.80	49.38	56.27	52.13
BiT	Frozen Backbone	52.47	52.13	48.96	58.82	53.10
	Fine Tuning	51.20	55.83	54.80	61.49	55.83
	From Scratch	52.18	47.70	47.77	51.24	49.22
Swin Tiny	Frozen Backbone	54.79	56.17	54.43	57.04	55.61
	Fine Tuning	56.11	**62.59**	55.86	**65.79**	**60.09**
	From Scratch	51.47	48.07	49.75	48.21	49.38
ConvNext	Frozen Backbone	48.44	51.56	51.03	54.48	51.88
	Fine Tuning	55.46	58.68	56.40	58.82	57.34
	From Scratch	52.46	49.31	49.08	51.64	50.62

Table 7: AP (in %) for different models and training methods across Deepfakes, Face2Face, FaceSwap, and NeuralTextures datasets (training set: ProGAN and CelebA-HQ-img).

The models didn't achieve very good predictions on the test sets, the classification of these images was also explored by different augmentations, results are available on table 8. It appears as if the pre-processing does not have as clear of an influence as it had on the images generated by GANs and DMs, but is not surprising given the difference in the data.

Percentage transformed		Testing Dataset				Average
Compression	**Blurring**	**Deepfakes**	**Face2Face**	**FaceSwap**	**NeuralTextures**	
-	-	52.77	55.18	52.28	56.35	54.15
20%	-	51.55	**56.63**	53.56	57.59	**54.83**
40%	-	52.83	52.14	50.77	**57.72**	53.37
60%	-	52.21	54.12	54.36	56.32	54.25
80%	-	49.18	52.86	50.93	55.93	52.23
100%	-	**53.87**	53.93	51.12	58.08	54.25
10%	10%	49.88	54.24	**54.40**	55.10	53.91
-	20%	51.04	53.51	51.53	57.34	53.86
-	40%	49.70	51.73	53.19	52.15	51.69
-	60%	48.42	52.74	52.30	54.79	52.06
-	80%	48.17	51.42	50.58	56.21	51.60
-	100%	50.52	53.16	50.34	56.58	52.65

Table 8: AP (in %) for Swin Tiny (batch size 128) trained on ProGAN with different augmentations and tested on FaceForensics++ test sets.

To explore this dataset further, trainings using a swin tiny model were conducted on each FaceForensics++ training sets and the generalization to other types of fake images was tested, the results are stored in table 9. The model that was trained on the deepfakes achieved accurate predictions on the images generated by GANs. It also showed slightly lower performances on the DDIM and DDPM test sets but performed well on PNDM and LDM.

Training Dataset	Testing Dataset							Average
	ProGAN	**StyleGAN**	**VQGAN**	**DDIM**	**DDPM**	**PNDM**	**LDM**	
Deepfakes	**100.00**	**96.99**	**100.00**	**87.53**	**83.21**	**94.60**	**98.65**	**94.14**
Face2Face	65.85	64.68	65.85	52.43	55.34	50.51	55.65	58.04
FaceSwap	48.37	47.14	48.37	72.36	64.79	69.05	63.49	59.37
NeuralText	59.15	51.45	59.15	44.62	41.74	47.63	55.17	51.13

Table 9: AP (in %) for swin tiny trained trained on different datasets and tested on various GAN and DM datasets.

Table 10 presents the results obtained on the FaceForensics++ test sets, this table shows that when not training on the same type of deepfakes, the predictions of the model are not very accurate, especially for the FaceSwap images.

Training Dataset	Testing Dataset				Average
	Deepfakes	**Face2Face**	**FaceSwap**	**NeuralText**	
Deepfakes	**99.79**	71.24	41.57	74.95	71.39
Face2Face	78.86	**99.70**	50.11	62.91	**72.40**
FaceSwap	63.06	75.21	**99.84**	51.36	72.37
NeuralText	81.82	63.99	44.55	**98.39**	72.19

Table 10: AP (in %) for swin tiny trained on different datasets and tested on FaceForensics++ datasets.

Additional trainings were performed to see the effect of training on multiple types of fake images, table 11 presents the results of these trainings on the GAN and diffusion-generated images, and table 12 presents the results on the FaceForensics++ dataset. A first observation is that models trained on ProGAN perform significantly better on the synthesized images.

Training Dataset	Testing Dataset							Average
	ProGAN	StyleGAN	VQGAN	DDIM	DDPM	PNDM	LDM	
FFpp1+FFpp2+FFpp3	49.71	44.28	49.71	76.50	74.94	75.08	59.45	61.70
FFpp1+FFpp2+FFpp4	65.50	54.37	65.50	41.64	41.84	40.68	56.95	51.58
FFpp1+ProGAN	**100.00**	96.99	100.00	87.53	83.21	94.60	98.65	93.72
FFpp3+ProGAN	**100.00**	**97.70**	**100.00**	**93.42**	**90.53**	**98.13**	**99.89**	**96.63**

Table 11: AP (in %) for swin tiny trained trained on different datasets and tested on various GAN and DM datasets. FFpp1,FFpp2,FFpp3,FFpp4 correspond to Deepfakes, Face2Face, FaceSwap, and NeuralTextures in the same order.

The AP obtained on the NeuralTextures test set by the model trained on deepfakes and ProGAN shows that it is possible to reach certain level of precision in the predictions on fake images that were not generated using the same methods and invites for future exploration on the matter.

Training Dataset	Testing Dataset				Average
	Deepfakes	Face2Face	FaceSwap	NeuralText	
FFpp1+FFpp2+FFpp3	99.68	**99.73**	99.48	69.39	**92.07**
FFpp1+FFpp2+FFpp4	98.51	98.56	44.86	**97.74**	84.92
FFpp1+ProGAN	**99.78**	71.24	41.57	74.95	71.89
FFpp3+ProGAN	69.24	73.97	**99.69**	51.00	73.48

Table 12: AP (in %) for swin tiny trained on different datasets and tested on FaceForensics++ datasets. FFpp1,FFpp2,FFpp3,FFpp4 correspond to Deepfakes, Face2Face, FaceSwap, and NeuralTextures in the same order.

5.5 Using Model Ensembles to achieve better generalization

Model ensembles are a way to improve the predictions for the ConvNeXt and Swin tiny models, an ensemble of both of these models could achieve a higher score on the test sets. Table 13 presents the result of the isolated models, and the results of the averaging of their predictions, showing that using simple averaging of the predictions of two models that complete each other improves the predictions.

Type of Detector	GANs and DMs							Average
	ProGAN	StyleGAN	VQGAN	DDIM	DDPM	PNDM	LDM	
ConvNeXt	**100.00**	88.07	**99.99**	**100.00**	**100.00**	**100.00**	**100.00**	98.29
Swin tiny	**100.00**	**99.99**	**100.00**	90.21	91.29	99.85	99.88	97.31
Model Ensemble	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**

Table 13: AP (in %) for ConvNeXt and Swin tiny (training set: ProGAN and CelebA-HQ-img), and the model ensemble (obtained by averaging the predictions of both models). Tested on ProGAN, StyleGAN, VQGAN, DDIM, DDPM, PNDM, and LDM datasets.

Another observation that was made was that using a model ensemble could have a positive impact on the classification of other types of deepfakes. Table 14 shows that the results on the FaceForensics++ dataset when testing with a model ensemble can be slightly better than for the models by themselves, opening the door for future exploration on the method.

Type of Detector	FaceForensics++				Average
	Deepfakes	**Face2Face**	**FaceSwap**	**NeuralTextures**	
ConvNeXt	55.46	58.68	56.40	58.82	57.34
Swin tiny	56.11	62.59	55.86	**65.79**	60.09
Model Ensemble	**56.22**	**62.62**	**55.93**	65.78	**60.14**

Table 14: AP (in %) for ConvNeXt and Swin tiny (training set: CelebA-HQ, ProGAN), and the model ensemble (obtained by averaging the predictions of both models). Tested on FaceForensics++ datasets.

Since our models fine-tuned on the ProGAN dataset had very good generalization over entire face synthesis images, the focus is now to use those models and models trained on the FaceForensics++ dataset to try and improve the generalization to other types of deepfakes. Figure 13 presents a heatmap that shows the density of predictions for Swin Tiny models fine-tuned on two distinct training sets. This gives us motivation to use the prediction of those two models trained on two different dataset to try and improve the overall performance on all testing sets.

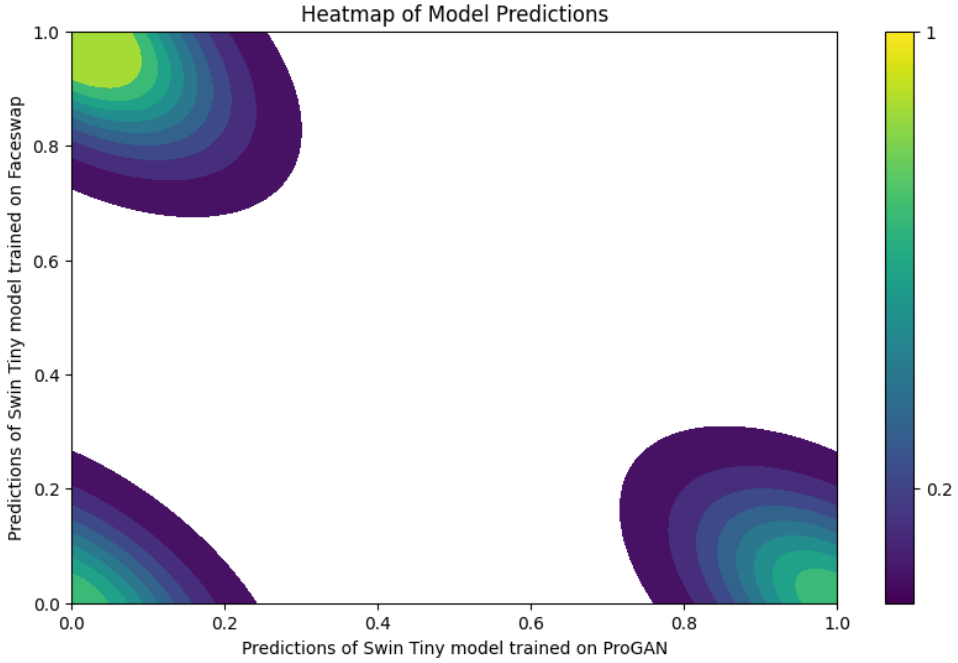

Figure 13: Heatmap of Model Predictions of two Swin Tiny models trained on FaceSwap (FaceForensics++) and ProGAN, the color bar indicates the density of the predictions on the ProGAN and FaceSwap validation set.

This figure shows us that models either do or don't agree confidently that an image coming from a different distribution is fake. A simple averaging of the predictions would allow us to separate the images.

The results presented below are the predictions made by a swin tiny model trained on FaceSwap (FaceForensics++), by another swin tiny model trained on ProGAN, the Model Ensemble that takes the average of the predictions of both models and a swin tiny model trained on both of the mentioned training sets. Table 15 presents the results on the images generated by GANs and DMs, table 16 presents the ones obtained on the FaceForensics++ test sets. On table 15, a first observation is that the swin tiny model trained on ProGAN performed better than the others and the model trained using both ProGAN and FaceSwap also performed well. Another observation that is made is that the average of the prediction didn't work as good as expected.

Type of Detector	GANs and DMs							Average
	ProGAN	StyleGAN	VQGAN	DDIM	DDPM	PNDM	LDM	
Swin tiny FaceSwap	48.37	47.14	48.37	72.36	64.79	69.05	63.49	59.37
Swin tiny ProGAN	**100.00**	**99.99**	**100.00**	90.21	**91.29**	**99.85**	99.88	**97.60**
Model Ensemble	**100.00**	98.41	**100.00**	76.73	69.44	96.65	98.52	90.21
Swin tiny ProGAN+Faceswap	**100.00**	97.70	**100.00**	**93.42**	90.53	98.13	**99.89**	97.24

Table 15: AP (in %) for different types of detectors, tested on ProGAN, StyleGAN, VQGAN, DDIM, DDPM, PNDM, and LDM datasets.

Type of Detector	FaceForensics++				Average
	Deepfakes	Face2Face	FaceSwap	NeuralText	
Swin tiny FaceSwap	63.06	**75.21**	**99.84**	51.36	72.37
Swin tiny ProGAN	56.11	62.59	55.86	65.78	60.59
Model Ensemble	58.20	65.35	99.43	**65.85**	72.21
Swin tiny ProGAN+Faceswap	**69.24**	73.97	99.69	51.00	**73.47**

Table 16: AP (in %) for different types of detectors, tested on FaceForensics++ dataset (training set mentioned with model name).

An interesting thing observed on table 16 is that the model ensemble is the best performer on the NeuralTextures test set. This shows that in some scenarios, using models trained on different datasets works better than using a single model trained on both.

6 Conclusion and Future Work

The research done on the detection of diffusion-generated images offered valuable insights, particularly in enhancing generalizability across various generative models by using different architectures. However, an important thing to consider is the difference in format and size of the real and fake images used for training the models. This may have caused specific patterns to be learned and explaining the non-generalization to the deepfakes from the faceforensics dataset. This would explain why, when the additional tests conducted with models trained on the deepfakes of the faceforensics dataset, a very high level of generalization on images generated by GANs and diffusion models was observed.

Our analysis revealed that using large pre-trained vision models, such as ConvNeXt, provided improved generalization performance across different types of GANs and diffusion models, highlighting the importance of pre-trained feature extraction. It was also shown that by using a model ensemble, a perfect classification of seven generative models was obtained, using training data from a single generative model.

This study also opens possibilities for further exploration; a large number of advanced data augmentation methods could be explored such as MixUp or CutMix. They could offer additional insights into improving the classification robustness. On top of that, model ensemble with a set of orthogonally trained models on entire images, patches, and frequency transformations could be interesting to explore. Addressing computational efficiency through model quantization could also be crucial for particular applications where inference time is important as using these model ensembles significantly increases the time needed for predictions.

Future exploration could also focus on increasing dataset variability by including images generated by more models during training and testing, which could lead to more robust detection methods. Additionally, using data from one generative model for training and another for validation, known as domain adaptation, could be an interesting experiment.

Following these research directions can considerably improve the authenticity verification of generated images and other synthetic media, addressing the security threats and misinformation related to synthetic content spread.

7 References

References

[1] Hangbo Bao et al. *BEiT: BERT Pre-Training of Image Transformers*. 2022. arXiv: 2106.08254 [cs.CV].

[2] Zihang Dai et al. *CoAtNet: Marrying Convolution and Attention for All Data Sizes*. 2021. arXiv: 2106.04803 [cs.CV].

[3] Patrick Esser, Robin Rombach, and Björn Ommer. *Taming Transformers for High-Resolution Image Synthesis*. 2021. arXiv: 2012.09841 [cs.CV].

[4] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[6] Itseez. *Open Source Computer Vision Library*. https://github.com/itseez/opencv. 2015.

[7] Tero Karras et al. *Analyzing and Improving the Image Quality of StyleGAN*. 2020. arXiv: 1912.04958 [cs.CV].

[8] Tero Karras et al. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2018. arXiv: 1710.10196 [cs.NE].

[9] Alexander Kolesnikov et al. *Big Transfer (BiT): General Visual Representation Learning*. 2020. arXiv: 1912.11370 [cs.CV].

[10] Yuezun Li et al. *Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics*. 2020. arXiv: 1909.12962 [cs.CR].

[11] Luping Liu et al. *Pseudo Numerical Methods for Diffusion Models on Manifolds*. 2022. arXiv: 2202.09778 [cs.CV].

[12] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV].

[13] Zhuang Liu et al. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV].

[14] Yuhang Lu and Touradj Ebrahimi. *Towards the Detection of AI-Synthesized Human Face Images*. 2024. arXiv: 2402.08750 [cs.CV].

[15] TorchVision maintainers and contributors. *TorchVision: PyTorch's Computer Vision library*. 2016. URL: https://github.com/pytorch/vision.

[16] Sara Mandelli et al. *Detecting GAN-generated Images by Orthogonal Training of Multiple CNNs*. 2022. arXiv: 2203.02246 [cs.CV].

[17] Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. *Towards Universal Fake Image Detectors that Generalize Across Generative Models*. 2024. arXiv: 2302.10174 [cs.CV].

[18] Ilija Radosavovic et al. *Designing Network Design Spaces*. 2020. arXiv: 2003.13678 [cs.CV].

[19] Benjamin Recht et al. *Do ImageNet Classifiers Generalize to ImageNet?* 2019. arXiv: 1902.10811 [cs.CV].

[20] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV].

[21] Andreas Rössler et al. "FaceForensics++: Learning to Detect Manipulated Facial Images". In: *International Conference on Computer Vision (ICCV)*. 2019.

[22] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV].

[23] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[24] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: 2010.02502 [cs.LG].

[25] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[26] Hugo Touvron et al. *Training data-efficient image transformers distillation through attention*. 2021. arXiv: 2012.12877 [cs.CV].

[27] Sheng-Yu Wang et al. *CNN-generated images are surprisingly easy to spot... for now*. 2020. arXiv: 1912.11035 [cs.CV].

[28] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.

[29] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[30] Bichen Wu et al. *Visual Transformers: Token-based Image Representation and Processing for Computer Vision*. 2020. arXiv: 2006.03677 [cs.CV].

[31] Saining Xie et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. arXiv: 1611.05431 [cs.CV].

A Additional Methods for Model Ensembles

During the final stage of the project, new methods were explored to leverage the feature spaces learned by different models. Instead of using the final predictions given by the models, the objective was to utilize the feature space similarly to Ojha et al. [17]. The key difference from their work is that this implementation uses a feature space trained on fake images. The aim was to leverage the differences in features learned by the models to train a meta-model and identify a pattern to generalize the classification of fake images.

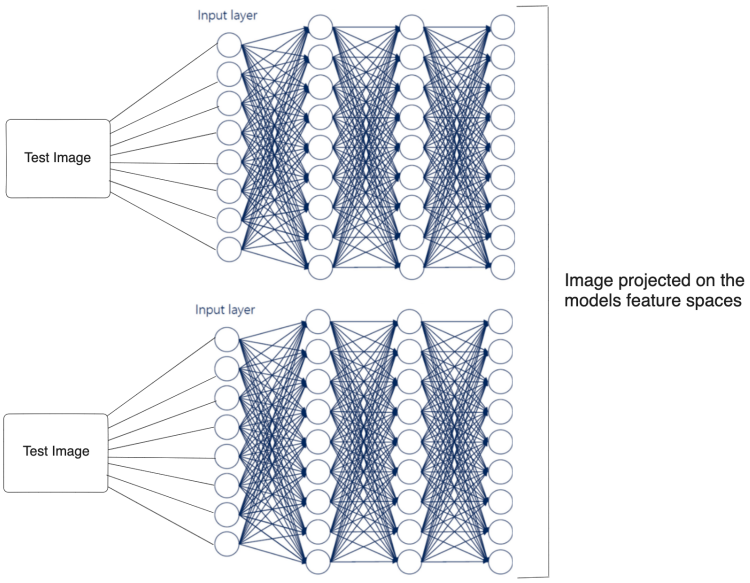

Figure 14: Representation of an image being projected onto the models' feature spaces. When using kNN, the distance to images from the training set is used to determine the class of the image tested.

The final classifiers proposed are a simple Linear Regression and a Nearest Neighbors (kNN) algorithm. It would also be interesting to train a classifier composed of a couple of fully connected layers for this task. Here is a detailed explanation of the two propositions:

1. The first method involves training a linear regression (meta-model) on the feature space of the training set. After training a set of models $M_1, M_2, \ldots, M_n$, the values predicted for the training set X_{train} are used to train a meta-model (meta-features F_{meta}). The meta-model M_{meta} is then trained using the meta-features F_{meta} and the true labels y_{train} from the training set. Stacking can be very useful to combine the strengths of multiple models to make a final prediction.

$$M_{\text{meta}} = \text{LinearRegression}(F_{\text{meta}}, y_{\text{train}})$$

   The meta-model is used to make the final predictions: $\hat{y}_{\text{test}} = M_{\text{meta}}(F_{\text{test}})$. For binary classification, a threshold of the continuous predictions $\hat{y}_{\text{test}}$ is used to obtain binary class labels:

$$\hat{y}_{\text{test}}^{\text{binary}} = \begin{cases} 1 & \text{if } \hat{y}_{\text{test}} \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

2. The second method proposed is to use *k Nearest Neighbors* (kNN) on the feature space of the trained models. Instead of giving the final predictions after passing through the fully connected layers, the images are projected onto the feature spaces of each model before

the classifiers. A distance metric (Manhattan, Euclidean, cosine, etc.) is then used to find the score attributed to the closest neighbors in the training set, and the average of those neighbors is used for the final classification of the image.