WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Bachelor Thesis

Is it possible to identify which AI Generator generated an image?

Laura Fabris

Subject Area: Information Business

Studienkennzahl: J033 561

Supervisor: Assist.Prof. PD Dr. Sabrina Kirrane

Date of Submission: 04. December 2024

Department of Information Systems & Operations Management, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR WIRTSCHAFTSINFORMATIK
UND OPERATIONS MANAGEMENT DEPARTMENT
OF INFORMATION SYSTEMS AND OPERATIONS
MANAGEMENT

Contents

List of Figures

List of Tables

Acknowledgements

First, I want to thank my supervisor Professor Sabrina Kirrane who supported me during the process of writing this thesis. She not only guided me during my research and the academic writing process but also mentored me during the coding phase. I am also deeply thankful for all the encouragement and support my family and friends gave me. And lastly, I want to thank the authors of the datasets used in this thesis for making them accessible and allowing its use for academic purposes.

Abstract

Nowadays images are not always created by humans, they can also be generated by AI. This leads to the question: Where does this image come from? being more and more important. In this thesis, we build a framework that not only determines if an image is AI-generated or not but also which model was used to generate the fake images. For this, we use the models ResNET-101, DenseNET-121, and MobileNETV3. We also analyze which models generate the images that are the hardest to detect as fake. In the end, we analyze how the models decide if an image is AI-generated or not. To do this we use SHapley Additive exPlanations (SHAP).

1 Introduction

In 2021 OpenAI made their Artificial Intelligence (AI) DALL-E public, which can generate images from text [41]. Currently, in 2024 we are on the third generation of DALL-E [42] and a lot of other AI art generators exist for example: Adobe Firefly [25], Generative AI from Getty Images [24] and Midjourney [39]. This shows how rapidly the field of generative AI has grown over the last few years. With this evolution, two issues are clearer, the first is that AI Art is harmful to artists [31] and the second is that training data and copyright is a tricky subject. This is the case because some of the training data is scraped from the internet [50]. Meta for example uses images from public accounts to train their AI [38]. At the Bloomberg TEC Summit [51], the Meta representative said: "*Our data advantage is really the publicized images that have been shared on Facebook and Instagram as well as the publicized text. So we do not train on private stuff we do not train on stuff that people shared with their friends. We do train on things that are public*". Because of this issue with the training data, it is important to not only differentiate between AI-generated images and real images but also which generator created the picture. The individual ethical reasons are not the only grounds to consider, which generator made an image but also the possibility of legal vulnerability. For example, The US Copyright Office is considering revising the law surrounding generative AI [38]. This is the motivation for the AI detector developed in the context of this thesis. This AI detector not only determines if an image is AI-generated or not but also goes more into detail about which model was used to generate the image. We use the models DenseNet-121, ResNet-101, and MobilenetV3 for the detection of AI-generated images. To determine which model generated an image we use two versions of MobileNETV3. We also analyze how the models decide if an image is AI-generated or not with the help of SHapley Additive exPlanations (SHAP).

1.1 Research question

AI-generated images bring issues with them. That is why the overall research question of the Bachelor Thesis is: How can we determine which model generated an image?

This question will be answered by analyzing the following sub-questions:

- How can we detect AI-generated images?
- What models generate images that are the hardest to identify?
- How does the algorithm decide whether an image is AI-generated?

1.2 Thesis Structure

The following is an overview of the rest of the thesis.

In **Chapter 2** we introduce the most important model types that are used for AI image generation. In this chapter, we also explain what techniques are currently used to detect AI-generated images. For that, we look at the literature in this field.

Chapter 3 introduces the framework of our AI detector and the deep learning models we use for it. In the second half of this Chapter, we explain the methodology we use and how we use it.

In **Chapter 4** we explain the datasets we use and how we preprocessed them.

In **Chapter 5**, we explain how we build the deep-learning framework of this thesis. First, we explain how the models are implemented. Then we look at the individual levels of the AI detector and evaluate them. At the end, we explain how we evaluated the overall framework and what the limitations of this framework are.

Chapter 6 explores how the models decide if an image is AI-generated or not. For that we use SHAP. In this chapter, we explain how we implemented SHAP and what the results are.

In **Chapter 7** we summarize everything and talk about how the research can be furthered in the future.

2 The Current State of the Art in AI Image Detection

In this chapter, we look at the current state of the art in AI image generation and detection and compare the different approaches. For Human AI detection we limited ourselves to papers about pictures not including videos, because we use images in the context of this thesis and videos could be detected differently from images. For papers about the ability of machines we also only use papers about image detection and not video detection and we only use papers where at least a part of the training dataset consists of images that are completely AI-generated. The reason for that is that there is a chance that a method that can detect if an image has part of itself changed by AI can not detect if an image is completely AI-generated.

2.1 The model types that are behind AI images

There are currently two algorithms that are used for the majority of AI image generation. Generative adversarial networks (GAN) and Diffusion models.

According to Ali et al. [2] and Baraheem et al. [4] a GAN consists of two neural networks: a generator and a discriminator. The generator is trained to generate images and the discriminator is trained to identify if an image is AI-generated. The discriminator checks the image created by the generator if the discriminator identifies the image as AI-generated the generator produces a new image. This is then repeated, till the discriminator accepts the image.

As explained by Dhariwal et al. [8] and Ho et al. [20] the first step to train a diffusion model is to make an image noisy. This is done by stepwise adding Gaussian noise to an image. The gradual adding of noise is the forward process of the training. When the image is noisy the reverse process starts. The reverse process works by learning variables and denoising a noisy image step by step.

According to Ahmad et al. [1] the strengths of GANs are their sample quality and inference speed. The weaknesses of GANs are their training stability and mode diversity. The strength of Diffusion models is, according to Ahmad et al. [1], their sample quality, training stability, and mode diversity, and their weakness is inference speed.

2.2 Humans' Ability to Detect AI-Generated Images

Pocol et al. [45] conducted a survey with 260 members which tried to differentiate between AI-generated images and real ones. There were 10 real

images and 10 AI-generated ones. The result is an average image classification accuracy of *0.61*. This is the average of the accuracy per image. The images in this study were all human faces and it was recommended to the participants to not overthink the answer. Also, heavily photoshopped images counted as real. Pocol et al. [45] concludes the survey results with: "*These results demonstrate that people are not good at separating real images from fake ones, easily allowing the propagation of false and potentially dangerous narratives.*" It was also analyzed why participants classified an image as fake and the number one keyword that was used is eyes. To get the keywords not only the answers of the correctly classified images were used so the results of the keywords could still lead to a false classification. However, the paper does analyze AI-generated eyes in detail and finds that the two signs for AI-generated eyes are heterochromia and white pixels around the eyes. Also, AI has problems generating realistic teeth.

A similar study was done by Groh et al. [17] in which they asked participants to find the image that was influenced by AI. In this study, two images were shown to the participants. One was real and the other one had something removed with the help of AI image manipulation. Before going to the next image the participants were told which one was AI manipulated. The result was a mean identification accuracy of *0.86*. The images are randomly selected and it was analyzed that 0.78 of the first images were classified correctly and *0.88* of the tenth images were classified correctly. This suggests that humans can learn how to identify AI-manipulated images. Important to mention is that not all participants classified 10 or more images because the number was not standardized but for the comparison between the first and tenth participants only the sample who answered at least ten was taken. Also, one unique IP address was counted as one unique participant. This suggests that it is easier for humans to detect images that are AI-manipulated than images that are AI-generated or that it is easier to detect the AI image out of two than to decide if one is AI-generated.

A study done by Rössler et al. [49] shows that these differences can also come from different quality and manipulation methods. In this study, the images were taken out from videos that have been manipulated at first with different methods and then post-processed to appear in different video qualities. In this study, 204 people participated most of them being computer science university students. This study showed each participant 60 images. The result shows that the quality has a big impact with the average accuracy for raw videos being *0.6896* and the accuracy for low-quality videos being *0.5873*. The accuracy of the different methods used to manipulate the images also shows a big difference, the reason for that is probably that some methods only add small semantic changes compared to other methods.

A study where the participants had no time limit to decide if the image was AI-generated or not was done by Lu et al. [36]. The participants consisted of 50 people, who achieved an accuracy of *0.6134*. Objects are classified as the least correct and multiperson pictures are the most correct.

2.3 Machines' Ability to Detect AI-Generated Images

In this section, we explain how machines can be utilized to identify AI-generated images. For that, we first look at the different deep learning methods and then at other machine methods. The results of all the machine methods can be found in Table 1. This Table includes the model that is used, the reference to the corresponding paper, the availability of the dataset, the type of images in the dataset, the accuracy, and the F1-score where available. Most papers use a dataset that is partly available because the real images are available and the fake images are produced specifically for their paper and then not made public. Sometimes the real part of an image is a subset of a publicly available dataset but it is not clear which images are included in the subset. The availability is set to partly as well in this case. If neither the accuracy nor the F1-score are available the paper is not included in the table. The methods are explained in more detail in the following subsections.

2.3.1 Deeplearing Methods

There are multiple ways in which deep learning can be used for AI image detection.

The Original Image The original images can be used for training deep learning models. This means that the images are only preprocessed with the preprocessing function of the model. This is for example done by Kusuma et al. [31], who trained the model specifically to detect AI-generated anime images. The results of this model can be seen in Table 1. The best accuracy of this method with this specialized dataset is *0.972*.

Baraheem et al. [4] use this method as well. Here the dataset spans multiple categories but the fake images are only generated by GAN architectures with different tasks like image-to-image synthesis and not by Diffusion models. The best accuracy in this experiment is *1.00* with EfficientNETB4. The results from the other models can be seen in Table 1.

A CNN was also trained by Wang et al. [60], who trained ResNET-50 to distinguish between AI and Real images. In this paper, Wang et al. [60] checked the influence of data augmentation on the robustness of this model by only training the model with images generated by one generator. We do

not have the accuracy but the average precision is between 0.882 and 1.00 for different models. Important to mention here is that the average precision of 1.00 was on the generator the model was trained with and that all the other generators are also GAN models. The results show that data augmentation improves generalization.

The DFT of an Image The discrete Fourier transform (DFT) of an image can be used to train a deep learning model as done by Alkishri et al. [3]. According to Alkishri et al. [3] and Guarnera et al. [18], the DFT uses a formula to make frequencies visible. These frequencies are different for AI-generated images and real images. Figure 1 shows the average of the DFT spectrum. This image is taken from [18]. DFT makes it possible to identify AI-generated images. The average of the DFT looks different for each generator. They mostly differentiate from each other with the position and strength of the light lines. The dataset Alkishri et al.[3] use consists of human faces and the best accuracy in this experiment is *0.99*. Table 1 compares different models used to differentiate AI-generated images and real images on different datasets.

It is not clear if Guarnera et al. [18] use the original images or the DFT of the images to train the deep learning model, because DFT is explained but never mentioned in their framework. The dataset in [18] consists of multiple categories including for example human faces and animals. The best result is an accuracy of *0.9904*.

The DCT of an image The Discrete Cosine Transform (DCT) also makes the frequencies of an image visible but in a different way. The DCT of AI-generated images shows a difference from Real images so Poredi et al. [46] train a self-build Convolutional Neural Network (CNN) and reach an accuracy of *0.9688*. The goal of this work is to make a model that detects AI images specialized for Social Media content.

Inter-pixel correlation To make sure the deep learning model generalizes better Zhong et al. [62] train it with the Inter-pixel correlation. This is done by using a method Zhong et al. [62] call *Smash&Reconstruction*. What this method does is shown in Figure 2 which is taken from [62]. The images are smashed and then the parts of the image are split into poor texture and rich texture. Out of the original image, two reconstructed images are produced one with rich texture and one with poor texture. This way the semantic information of the image is broken, which makes sure the model does not train on the semantic information. High pass filters are then put over the

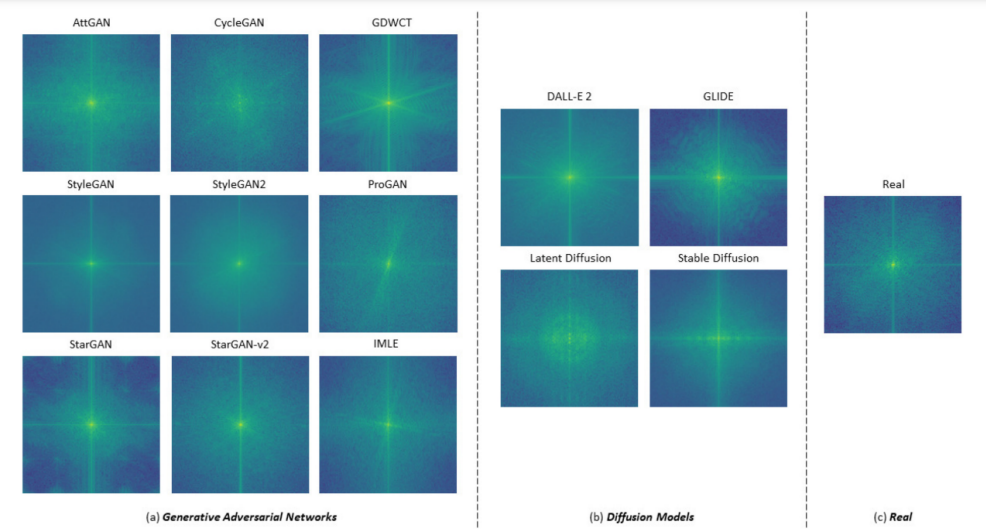

Figure 1: Average of the DFT spectrum [18]

reconstructed image. The images are used to train a self-built neural network. This method reached an average accuracy of *0.8985* over 17 individual datasets each generated by one generator. This method performs this well even though the images that were used for training only consisted of images generated by ProGAN. The Accuracy of the ProGAN dataset was even *1.00*. This proves that this method is very good at generalizing while training.

2.3.2 Other Methods

1D Power Spectrum A method that builds on the DFT method is used by Durall et, al. [10], who form the Azimuthal Average of the DFT of an image. This Average produces the 1D Power Spectrum which is then used to train SVM, Logistic Regression, and K-Means with different datasets. SVM and Logistic Regression reach a test accuracy of up to *1.00* and K-Means performs the worst but still reaches up to *0.96* test accuracy.

[1]There are currently only subsets of this dataset available. But [5] states: "Due to multiple reports of inconsistent metadata/files, Danbooru2021 has been taken offline until I either figure out the problem or make a fresh release". So the dataset could become available again

Model	**Reference**	**Available**	**Type**	**Accuracy**	**F1-score**
DenseNet-121	[18]	partly	mult. cat. (multiple categories)	0.9768	0.92
DenseNet-121	[3]	available	human faces	0.92	0.92
DenseNet121	[4]	partly	mult. cat.	0.97	0.97
EfficientNET-b4	[18]	partly	mult. cat.	0.8996	0.76
EfficientNetB4	[4]	partly	mult. cat.	1.00	1.00
EfficientNET-b7	[18]	partly	mult. cat.	0.9661	0.89
InceptionV3	[4]	partly	mult. cat.	0.98	0.98
InceptionResNetV2	[4]	partly	mult. cat.	0.96	0.96
K-Means	[10]	partly	human faces	0.96	-
Logisitc Regression	[10]	partly	human faces	1.00	-
MaxViT	[4]	partly	mult. cat.	0.89	0.89
MixConv	[4]	partly	mult. cat.	0.94	0.94
MobileNetV2	[31]	partly	Anime	0.968	0.971
MobileNetV3	[31]	partly	Anime	0.972	0.974
ResNet-18	[18]	partly	mul. cat.	0.965	0.89
ResNet-34	[18]	partly	mul. cat.	0.9761	0.92
ResNet-50	[18]	partly	mul. cat.	0.9818	0.94
ResNet50	[4]	partly	mult. cat.	0.91	0.91
ResNet-101	[18]	partly	mul. cat.	0.9893	0.96
ResNet101	[4]	partly	mult. cat.	0.95	0.95
ResNet152	[4]	partly	mult. cat.	0.92	0.92
ResNEXT-101	[18]	partly	mult. cat.	0.9729	0.91
Self-Build	[62]	partly	mult. cat.	0.8958	-
Self-Build	[46]	partly	mult. cat.	0.9688	-
SVM	[10]	partly	human faces	1.00	-
VGG16	[3]	available	human faces	0.99	0.99
VGG19	[4]	partly	mult. cat.	0.94	0.94
ViT-B16	[18]	partly	mult. cat.	0.9904	0.96
Xception	[4]	partly	mult. cat.	0.97	0.97

Table 1: Comparative Analysis of Model Results AI vs Real

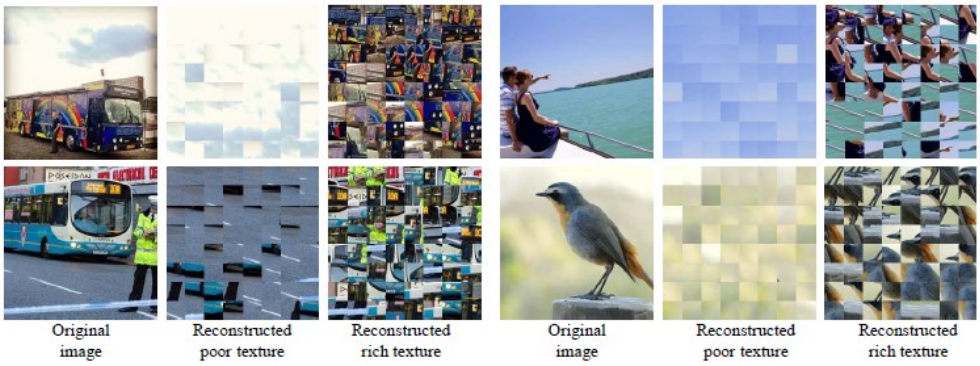

Figure 2: Illustration of *Smash&Reconstruction*[62]

2.4 Detection of the Generator that generated an image

It is also possible to detect which model was used for generating the image as done by Guarnera et al. [18]. This approach works by having multiple levels. The first level is trained to detect if the image is AI-generated or not. The second is to differentiate between GAN-generated images and Diffusion model-generated images. The third level is then to detect what model was used for generating the picture. This level consists of two models one of which differentiates between all the GAN models and one that differentiates between all the Diffusion models. The team achieved an accuracy of over 90% with these architectures overall and at the individual levels. Table 2 shows the results of the overall framework. This means the test data runs through all levels. The data that was used in this paper spans multiple categories. In this case, the models were all trained on the same data.

Model	Reference	Available	Type	Accuracy	F1-score
ResNet-18	[18]	partly	mult. cat	0.9606	0.98
ResNet-34	[18]	partly	mult. cat	0.9621	0.98
ResNet-50	[18]	partly	mult. cat	0.9601	0.97
ResNet-101	[18]	partly	mult. cat	0.9782	0.98
ResNEXT-101	[18]	partly	mult. cat	0.952	0.98
DenseNet-121	[18]	partly	mult. cat	0.972	0.99
EfficientNET-b4	[18]	partly	mult. cat	0.8794	0.9
EfficientNET-b7	[18]	partly	mult. cat	0.9341	0.95
ViT-B16	[18]	partly	mult. cat	0.9533	0.98

Table 2: Comparative Analysis Results Generator Differentiation

3 Research method

In this chapter, we explain the architecture of our AI detector including the models that we use. We also explain the methodology we use to build an AI detector that can differentiate between the models that generated the images.

3.1 Framework

The Framework is the construct of deep learning models that differentiates between real images and the different generators.

3.1.1 Models

The AI detector built in the context of this thesis consists of deep learning models. The type of deep learning model we use are CNNs. As explained by O'Shea et al. [43] a CNN consists of self-optimizing neurons that learn. The use cases of CNNs are mostly pattern recognition within images. The unique feature of CNNs is that their neurons are organized in three dimensions: height, width, and depth, which is the reason why this type of Neural Network is used for image-centered use cases. An example of a CNN architecture can be seen in Figure 3 which is taken from [43]. The convolutional layers that can be seen in the Figure is the layer that learns. The task of a pooling layer

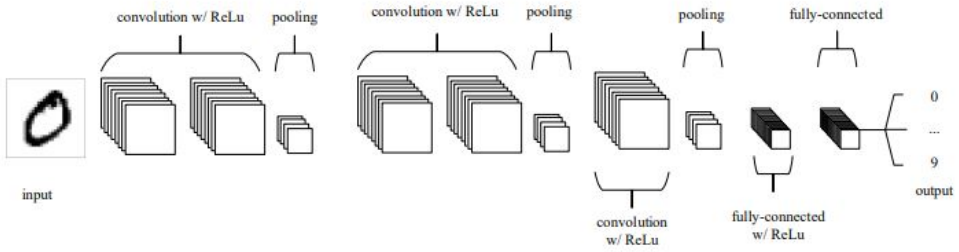

Figure 3: Example of a CNN architecture [43]

is to reduce the dimensionality by scaling down the activation map without losing depth volume. The important part of the fully-connected layers is that each neuron is connected to all neurons of the preceding and the following layer, but not connected to a neuron on the same layer.

MobileNetV3 MobileNet is a series of networks that were developed with the computational limitation of mobile devices taken into consideration [22]. As described by Howard et al.[21] the family of MobileNets consists of the versions MobileNet, MobileNetV2, MobileNetV3-Small, MobileNetV3-Large and MobileNetV4 [1]. Each generation improves the model in another way. MobileNet uses depthwise separable convolution. This enhances the efficiency. MobileNetV2 builds on this as explained in [21]: "*...depthwise separable convolution to substantially improve computation efficiency*". MobileNetV3 is built for the optimal accuracy and latency for mobile computer vision architectures. MobileNetV3 is the one used here because it performed the best in [31]. It is not clear whether the small or the large version is used in this paper, thus we are using both models here to compare them and see which one performs better for this use case.

ResNet-101 ResNet-101 is a residual network with 101 layers. Residual networks were developed to solve a big problem from deep learning models: after a certain number of layers, the accuracy of the models saturates at first and then degrades. The reason for that is not overfitting but that if the input goes through too many layers information can get lost [23]. According to He et al. [19] Residual networks tackle this problem by adding identity mapping layers. The building block of residual learning works the following way: An identity or input is put through weight layers and in between the layers in the

[1] is the newest generation and came out this year (2024) [47].)

ReLu activation function. The ReLu activation function is f(x) = max(0,x) [35]. This means that the negative input of this function is always set to at least zero so the output can never be negative. The output of these layers is then added to the original identity. This addition to the original identity is what differentiates residual Networks from other deep learning models.

DenseNet-121 DenseNet-121 is a Densely Connected Convolutional Network (DenseNet) with 121 layers. According to Huang et al. [23], DenseNet wants to solve the vanishing gradient problem from deep learning models but it uses a different technique than ResNet. DenseNets solution is to not only use the output of the prior layer of the model but all the layers coming before and its feature map. The difference between ResNets and DenseNets method is that ResNets uses summation to combine features and DenseNet concatenates them.

3.1.2 Levels

To not only differentiate between AI-generated images and real images but also between the different generators the framework is split into multiple levels which can be seen in Figure 4. This structure is the same as Guarnera et al. [18] use. The framework consists of 3 levels and the first one is AI vs Real, which identifies the AI-generated images. Level two then further differentiated the AI-generated images into images generated by GAN or Diffusion models. Level three then splits these two into the individual models. We train four different models on level 1 and two models for levels 2 and 3.

3.2 Action Research

The work described in this thesis is guided by the Action Research [6] methodology, which distinguishes itself because of its flexibility. According to Checkland et al. [6] this flexibility is because in Action Research three elements can change during the research. These elements are the area of concern, particular linked ideas, and the methodology. These elements need to be defined before the process starts so that it is possible to analyze should there be changes. In our case, the elements do not change during the process, because change is not needed to answer our research question. Our definition for the particular linked ideas are deeplearning, AI detection, and AI image generation, and our definition for the area of concern is the identification of AI-generated images. Our methodology consists of a typical AR cycle. As explained by Dickens et al. [9] the steps of this cycle can vary depending on the project and do not always need to be followed statically as explained by

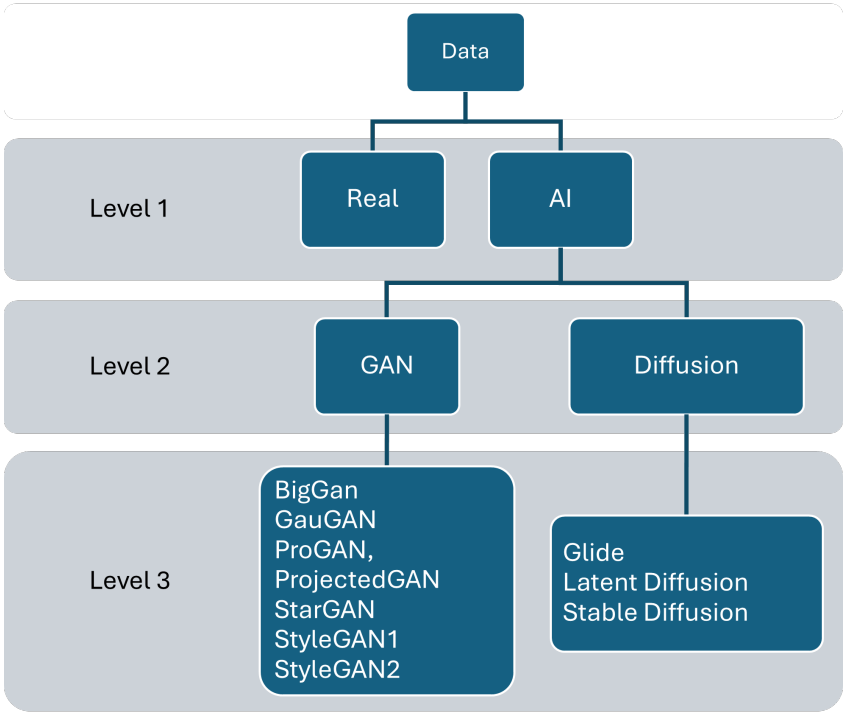

Figure 4: Levels of the Framework

Dickens et al. [9] *"Rather, it (action research) can go forward, backward, and in all directions at once"*. Our approach is inspired by [15] and [34] with four steps which are:

1. Plan
2. Act
3. Analysis
4. Reflect

These four steps are followed to build the overall framework and are done on a smaller level in the action step for each level of the framework as seen in Figure 5. The cycles are repeated till there seem to be no more ways to improve the results with the resources of this bachelor thesis.

3.2.1 Plan

At first, the model framework is planned. The model consists of three levels:

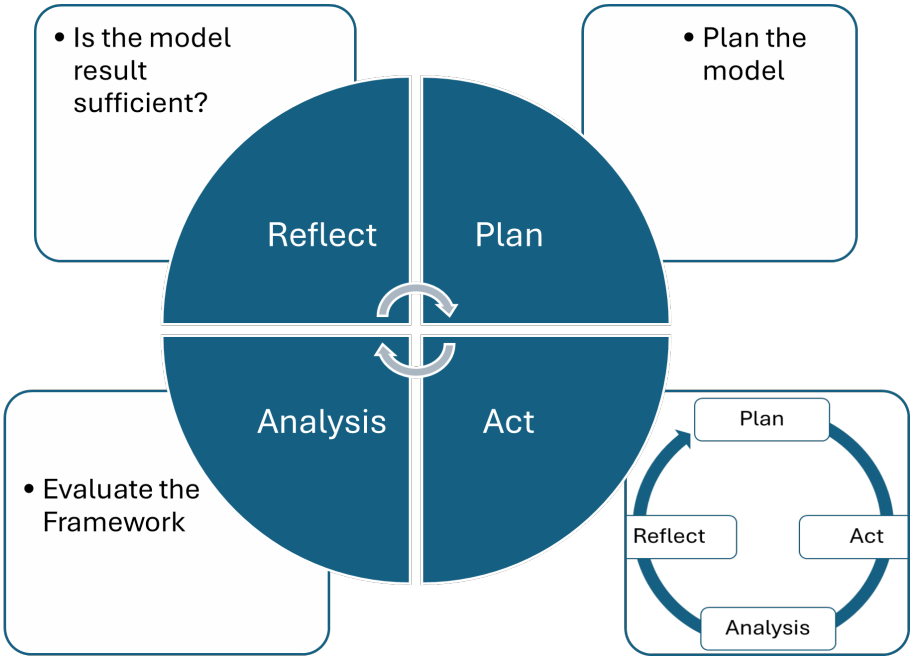

Figure 5: Methodology

1. AI vs Real
2. GAN vs Diffusion
3. Identify the Models

After we know which data is necessary we plan which data is used and how it is preprocessed. Then we plan how each individual level is built. The levels consist of deep learning models. Four models are chosen and compared to see which performs the best. The first two models are types of MobileNetV3 which are also used in [31]. Version 3 performs the best in identifying the AI images, but it is not mentioned if the Small or the Large version is used, so we use both. The other model that we use is DenseNet-121 which performed best in the overall framework in [18] with the metrics precision, recall, and F1. The fourth model that we use is ResNet-101. This model performed the same in recall as DenseNet-121 and the best in accuracy of the overall framework also in [18]. The last part of the first planning sequence in the cycle is planning the evaluation process. The metrics that are used for the evaluation are accuracy, precision, recall, and F1-score. Also, a confusion matrix is built to compare the models. The reason we use these metrics is that they have been used in multiple papers including [31], [18], [4] and [3].

3.2.2 Act

This step consists of smaller cycles for each level. At first, the model is **planned** in more detail and fitted to the individual model. Then the model is implemented. This is the **act** step. The model is evaluated in the **analysis** step. The metrics that are used are as aforementioned accuracy, precision, recall, F1-score, and a confusion matrix. Then the cycle is **reflected** on. In the reflect step it is decided if the cycle is repeated or if the result is satisfying. This decision takes two things into consideration. First of all, if the metrics are over a certain threshold in our case 70 %, and secondly if there seems to be a way to improve the model. The accuracy can for example be high because of an imbalanced dataset where all the images are classified in the same category. So the cycle begins again by planning how the dataset could be improved. We repeat each of the cycles for the models between 10 and 20 times.

3.2.3 Analyis

The analysis consists of the evaluation of the overall framework. The levels of the framework are already individually evaluated in the act cycle so that is not repeated here. We used the same metrics as before so accuracy, precision, recall, F1-score, and a confusion matrix.

To calculate the metrics we use the sklearn.metrics library [44]. For the binary classifications, the functions accurracy_score, precision_score, recall_score, and f1_score are used. The variable names that are used in the formulas are tp for true positives, fp for false positives, tn for true negatives, and fn for false negatives. The following formulas are analyzed with the function inspect.getsource [44].

Accuracy = (tp + tn) / (tp + tn + fp + fn)
Precision = tp / (tp + fp)
Recall = tp / (tp + fn)
F1-Score = (2 * tp) / (2 * tp + fp + fn)

3.2.4 Reflect

In this step, the decision takes place if the cycle is repeated. The cycle starts from the beginning if the results of the overall framework are not sufficient. The results are not enough if the accuracy is not at least over 70 % percent. Accuracy is not the only thing taken into consideration. Recall, precision, and the F1-score also need to be over 70%. The confusion matrix is to see if there are any anomalies in the result for example an imbalanced dataset. We repeat the overall cycles three times for MobileNETV3-Large and two times

for MobileNETV3-Small. There are not that many repetitions necessary here because the individual levels are already optimized as well as possible in the Act step.

4 Data and Preprocessing

In this chapter, we explain, which datasets we used and how we preprocessed them. We used two datasets to train the models both of them sourced from Kaggle. The first one is ai-generated-images-vs-real-images (AIvsReal)[61]. The dataset consists of a total of 60,000 images. Half of them are AI-generated and the other half are real images. The real images of this dataset are from Pexels [56], Unsplash [57], and WikiArt [58] and the AI-generated ones are from Stable Diffusion [55], MidJourney [39], and DALL-E [54]. The license is MIT. The license text can be found in Appendix A. On Kaggle the dataset is split into train and test data with a ratio of 80% and 20%. We put all the data together in two folders AI and Real for further preprocessing. The second dataset is called "ArtiFact: Real and Fake Image Dataset" (ArtiFact) [48]. The dataset consists of a total of 2,496,738 images. This splits into 964,989 real images and 1,531,749 fake images. The fake images were generated by 25 generators and the real images were sourced from 8 different sources. The dataset includes images from the categories: human, human faces, animal, animal faces, places, vehicles, art, and other real-life objects. The resolution of the images is 200x200. We do not use the whole dataset because of computational limitations. The generators we use in this thesis are BigGan, GauGAN, Glide, Latent Diffusion, ProGAN, ProjectedGAN, Stable Diffusion, StarGAN, StyleGAN1, and StyleGAN2. Table 3 contains the models we use, whether it is GAN or Diffusion model, and what license they have. We use a random sample of 7,000 images from all the generators, except for Glide because we only have 1,000 images available. 66,929 random images of the real categories are chosen to keep the ratio. The images are picked randomly by the random.sample() function from the random library, which is a Python library that includes random functions for integers and sequences [14]. The datasets we use as the real images are subsets of "ArtiFact: Real and Fake Image Dataset" [48]. The dataset is already split into these subsets which include images from the datasets FFHQ, ImageNet, and AFHQ. We form our subsets of these subsets by using the random.sample() function. The corresponding licenses of these subsets are for FFHQ Creative Commons BY-NC-SA 4.0 license, for ImageNet Non-Commercial, and for AFHQ Creative Commons Attribution-NonCommercial 4.0 International Public. The texts to the licenses can be found in Appendix A.

Preprocessing The preprocessing for the AIvsReal model works by cutting down the dataset. This is done by taking 7,000 images from each generator except for Glide which only has 1,000 images. The images are chosen randomly. The images are then split into the ratios 70% training data, 15%

Method	Model Type	License
BigGan	GAN	MIT License
GauGAN	GAN	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License
Glide	Diffusion model	MIT License
Latent Diffusion	Diffusion model	MIT License
ProGAN	GAN	Attribution-Noncommercial 4.0 International
ProjectedGan	GAN	MIT License
Stable Diffusion	Diffusion model	Apache-2.0 License
StarGAN	GAN	MIT License
StyleGan1	GAN	Creative Commons Public Licenses
StyleGan2	GAN	Nvidia Source Code License

Table 3: Summary of Models

validation data, and 15% test data. The split is done using Algorithm 1. After the split, the data is put in their respective folder in the folder structure which can be seen in Figure 6. The images are moved with the os library, which enables us to work on the operating system, and the pathlib library, which provides functions to present filesystem paths correctly [12], [13].

We then resize the images to the size 224x224 for the models ResNet-101 and DenseNet-121. We do not resize the images for the MobileNET models, because it is not necessary for the further preprocessing and training of the model. The individual images are preprocessed with the integrated preprocessing functions of the models.

Preprocessing GANvsDiffusion The preprocessing for the GANvsDiffusion level consists of cutting down the GAN images to 15,000 to have the same number of images generated by Diffusion models and images generated by GAN. The images are chosen with the random.sample() function. The images are taken from the already-split folders so the ratios remain the same with 70% training data 15 % validation data and 15% test data. The images are distributed to the different folders shown in Figure 7.

Preprocessing Differentiation GAN The preprocessing for the Differentiation between the GAN images consisted of picking 7,000 images per generator dataset. This is done with the random.sample() function.

Algorithm 1 Split Algorithm

function SPLIT(path)

$train \leftarrow 0.7$
$validation \leftarrow 0.15$
$test \leftarrow 0.15$

$images \leftarrow \textbf{add}(List\ of\ Images\ in\ the\ folder)$
$length \leftarrow \textbf{length}(images)$

$train\_num \leftarrow length * train$
$test\_num \leftarrow length * test$
$val\_num \leftarrow length * validation$

$train\_list \leftarrow \textbf{random}(train\_num\ images\ from\ images)$
$images \leftarrow images - train\_list$

$validation\_list \leftarrow \textbf{random}(val\_num\ images\ from\ images)$
$images \leftarrow images - val\_list$

$test\_list \leftarrow \textbf{random}(test\_num\ images\ from\ images)$
$images \leftarrow images - test\_list$
return $train\_list\ test\_list\ val\_list$

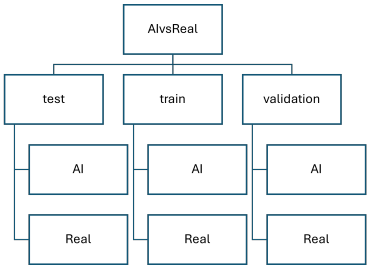

Figure 6: Folder structure AIvsReal

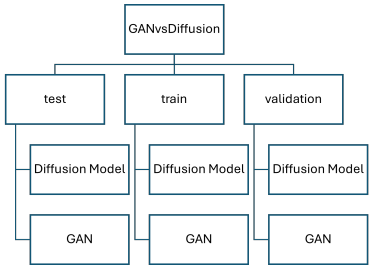

Figure 7: Folder structure GANvsDiffusion

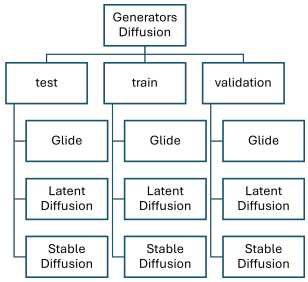

Figure 8: Folder structure Generators Diffusion Models

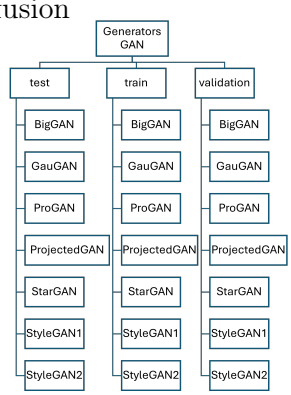

Figure 9: Folder structure Generators GAN Models

The images are then put into their respective folder which can be seen in Figure 9 again with the ratios 70-15-15.

Preprocessing Differentiation Diffusion Models For the Diffusion model differentiation, we have three Diffusion models, Glide, Latent Diffusion, and Stable Diffusion. The dataset only includes 1,000 Glide images. We used 7,000 pictures each for Latent Diffusion and Stable Diffusion to keep the training data big enough. The 7,000 images are chosen randomly again. The images are split in the ratio 70-15-15 again into the folders that can be seen in Figure 8

5 Building the overall Framework

In this chapter, we explain how we implemented the models and analyze the results.

5.1 Implementation of the models

The implementation is done on two laptops, because there is nothing available with more computational power, in the context of this thesis. One of the laptops has an Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz 2.80 GHz processor and has an installed RAM of 8.00 GB. The other laptop has an Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz processor and an installed RAM of 16.00 GB. The models are implemented using Keras, which is a deep-learning API [30]. It is used on top of TensorFlow, an open-source deep-learning software library from Google [16]. Keras helps to define, train, and deploy machine learning models [16].

At first, we clarify how the data batches are built. We use the class mode categorical and shuffle the train batches and the validation batches. The test batches are not shuffled. Then the pre-trained model is loaded. The model is not pre-trained for our specific case but on the ImageNET dataset [59]. We use the softmax activation function because it simplifies using SHAP and it makes adapting the code for the other levels easier. We compared it with binary classification and the sigmoid activation function, but it did not influence the result. The categories for the first level are AI and Real, for the Second level GAN and Diffusion, and for the third level the generators. We train all the layers of the models and use the categorical crossentropy loss function with the accuracy metrics. Then we train the model for our specific data and use case. The variables that we change during the training are:

learning rate The learning rates are chosen in comparison to each other based on which performs the best. They vary for the different models and datasets. The learning rate for the AIvsReal dataset is 0.0001 for the MobileNETV3-Large model. ResNet-101 and DenseNet-121 are mostly trained with a training rate of 0.00001 and then changed to 0.000001 for the last epochs of training. The learning rate for the ArtiFact dataset is 0.00001 for both models.

batch size We change the batch size during training depending on how the accuracy shifts, we use a batch size between 1 and 10. MobileNET on the AIvsReal dataset works with bigger batch sizes ranging between 4 and 10 than ResNet-101 and DenseNet-121 which both use a batch size

of 3. For the ArtiFact dataset, we use a batch size of 3 which is smaller than the batch size for the AIvsReal dataset.

steps per epoch The steps per epoch are adapted to the accuracy during training with values between 18 and 50. The steps per epoch are higher for ResNet-101 and DenseNet-121 with 20 steps per epoch than for MobileNETV3-Large which used 18 steps per epoch. For the ArtiFact dataset, we use a higher steps per epoch value of 50 than for the AIvsReal dataset.

epochs The epochs also differentiate between models and datasets. For MobileNETV3-Large on the AivsReal dataset we need under 100 epochs and for the ResNet-101 and DenseNet-121, we need around 100 epochs. For the ArtiFact dataset, we need over 300 epochs for MobileNET models.

As seen above ResNet-101 and DenseNet-121 need values that need more computational power. The training of the ArtiFact dataset also needs values that lead to the necessity of more computational power. In our case, DenseNET-121 and ResNEt-101 are not learning the difference between AI and Real in the ArtiFact dataset. Why this could be the case is discussed in the next paragraph.

Discussion of the Differences in the Implementation. The reason why MobileNETV3-Large is more computationally friendly than the other models on the AIvsReal dataset is that this model is specialized to work on mobile devices whereas the other models are specialized in improving the performance not considering computational limitations. This is probably also the reason why the ArtiFact dataset does not work with DenseNet-121 and ResNet-101. As stated above this dataset needs values that are more computationally expensive. The combination of a dataset and models that are resource-intensive does not work with our limitations.

5.2 Levels of the Framework

To differentiate between the methods used for image generation and not just between AI and real we build a three-level framework which can be seen in Figure 4 and is explained in Chapter 3. The first level is AI vs Real. The following levels are to differentiate between GAN and Diffusion models. Level 3 depends on the result of level 2. There are two models for level 3. One that differentiates between all the GAN models and one that differentiates between all the Diffusion models.

Model	Dataset	Accuracy	Precision	Recall	F1-score
MobileNetV3-Large	AIvsReal	**0.924**	0.8950	**0.9607**	**0.9267**
DenseNet-121	AIvsReal	0.9052	0.8886	0.9267	0.9072
ResNet-101	AIvsReal	0.9111	0.9011	0.9236	0.9122
MobileNetV3-Large	ArtiFact	0.9238	**0.9169**	0.932	0.9244
MobileNetV3-Small	ArtiFact	0.8776	0.8745	0.8816	0.8781
DenseNET-121[2]	ArtiFact	0.4389	0.4557	0.6282	0.5282
ResNET-101[2]	ArtiFact	0.4630	0.4743	0.6816	0.5593
MobileNetV3-Large[2]	ArtiFact	0.4631	0.4770	0.7666	0.5881
MobileNetV3-Small[3]	AIvsReal	0.5061	0.5031	0.9878	0.6667
MobileNetV3-Large[3]	AIvsReal	0.5014	0.5007	0.9762	0.6619

Table 4: Results AIvsReal

5.2.1 AI vs Real

The first level of the overall framework distinguishes between AI-generated images and real images. We use two different datasets and 4 different models to compare how good the models are at detecting AI-generated images.

Evaluation and Results We evaluate the results by calculating the metrics of accuracy, precision, recall, and the F1-score. The values are calculated using the respective function from the sklearn.metrics library.

Table 4 shows the results of the models on the different datasets. The results are rounded to four decimal points. MobileNetV3-Large performs the best on the AIvsReal Dataset with an accuracy of 0.9240 and an F1-score of 0.9267. It is followed by ResNet-101 with an accuracy only slightly worse of 0.9111. DenseNet-121 performed the best in recall with 0.9167. When looking at the confusion matrices it is visible that both DenseNet-121 and ResNet-101 classify more fake images wrongly as real than real images wrongly as fake. This is in contrast to MobileNETV3-Large which classifies more real images as fake than fake images as real.

When the ArtiFact dataset is used MobileNETV3-Large performs better

[2]Trained on AIvsReal
[3]Trained on ArtiFact

Model	Accuracy	Precision	Recall	F1-score
MobileNETV3-Large	**0.9244**	**0.9377**	**0.9093**	**0.9233**
MobileNETV3-Small	0.9007	0.9230	0.8742	0.8980

Table 5: Results GANvsDiffusion

than MobileNETV3-Small. The large version achieves an accuracy of 0.9238. Both the MobileNETV3 version classified the wrong images pretty evenly between real and fake which can be seen in Figure 13 and Figure 14.

We also tested the models with the dataset it was not trained with. Here the results are between 0.4389 and 0.4631.

Discussion One reason MobileNetV3-Large could perform better than the others is that the models are trained on mobile devices and the MobileNet models are made to work on them. The results can differ depending on the dataset. The reason for that could be the diversity of the dataset.

There are multiple reasons why the model performs so badly on the test data of the dataset it was not trained on. One of them could be that the model learns patterns that are not connected to whether the image is AI or not. Another reason why the results are this bad could be that the fake images are generated by different models. The only model that is used in both datasets to generate images is Stable Diffusion but there could have been a different version used.

5.2.2 GAN vs Diffusion

The second level of the framework distinguishes between images generated by GAN and images generated by Diffusion models. We use MobileNetV3-Large and MobileNETV3-Small. DenseNet-121 and ResNet-101 are not used for this and the following level because of computational limitations.

As seen in Table 5 MobileNETV3-Large performs better in all the categories so if if the computational power is good enough MobileNETV3-Large should be used. MobileNETV3-Large achieves values better than 0.9 in all the categories whereas the recall of MobileNETV3-Small is 0.8742. This led to the F1-score of MobileNETV3-Small also only being 0.8980. The confusion matrices, that can be seen in Figure 15 and Figure 16 show that more GAN images are classified wrong as Diffusion models than Diffusion models being classified wrong as GAN images. This is the result of both models.

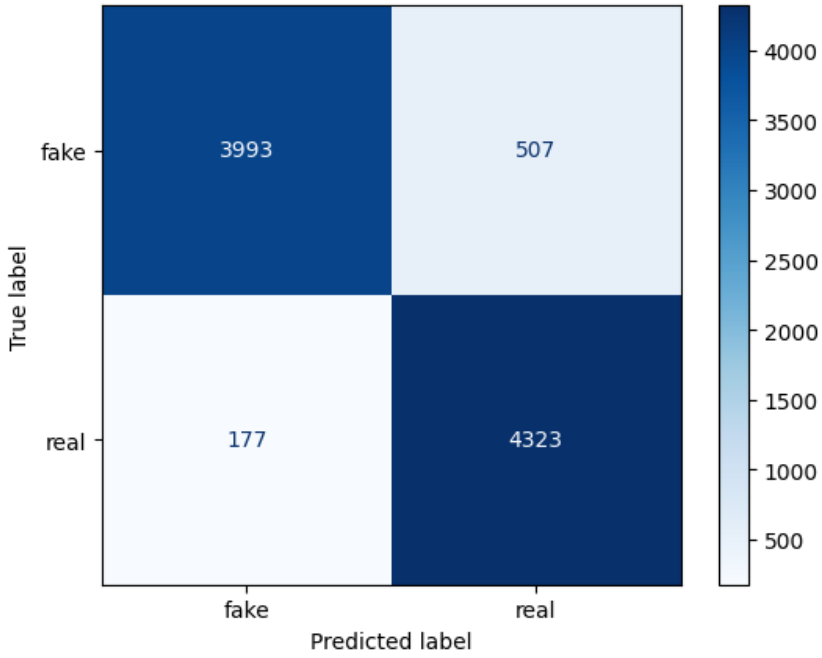

Figure 10: MobileNETV3-Large AIvsReal

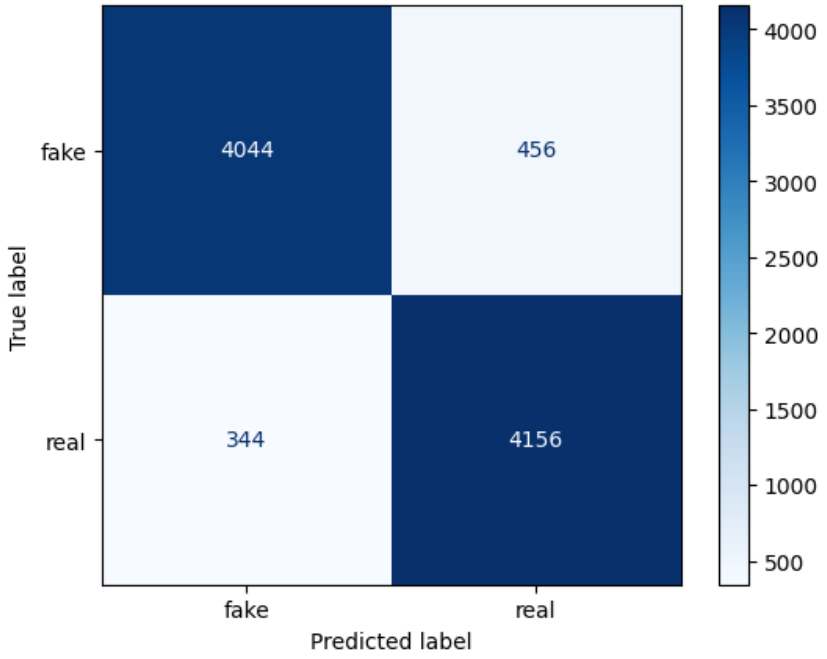

Figure 11: ResNet-101 AIvsReal

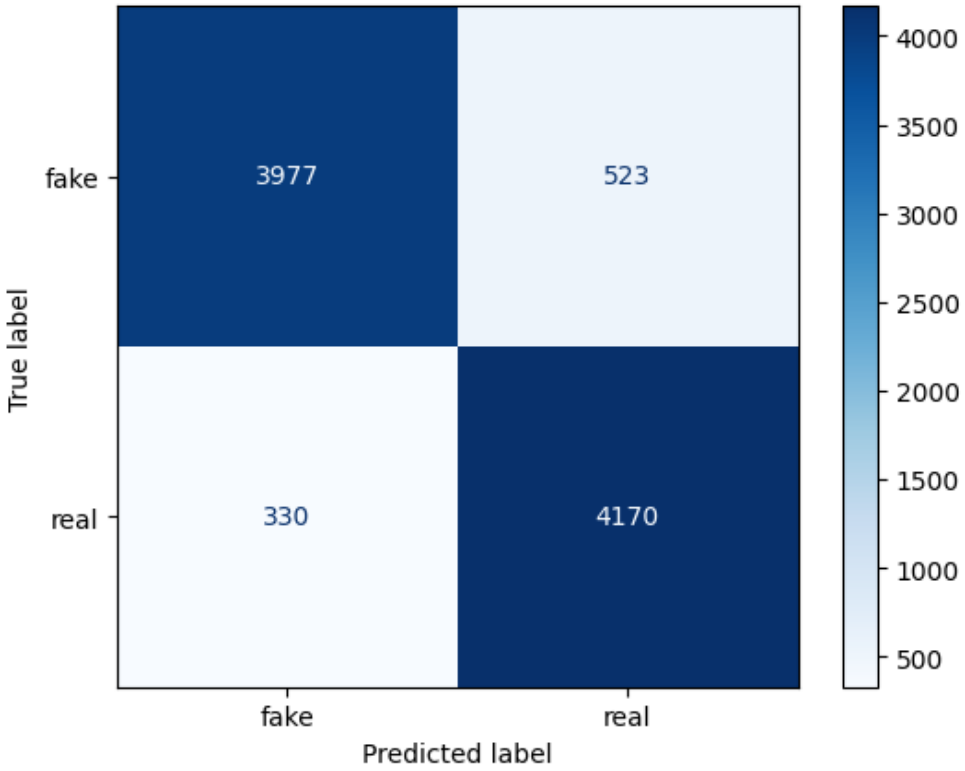

Figure 12: DenseNet-121 AIvsReal

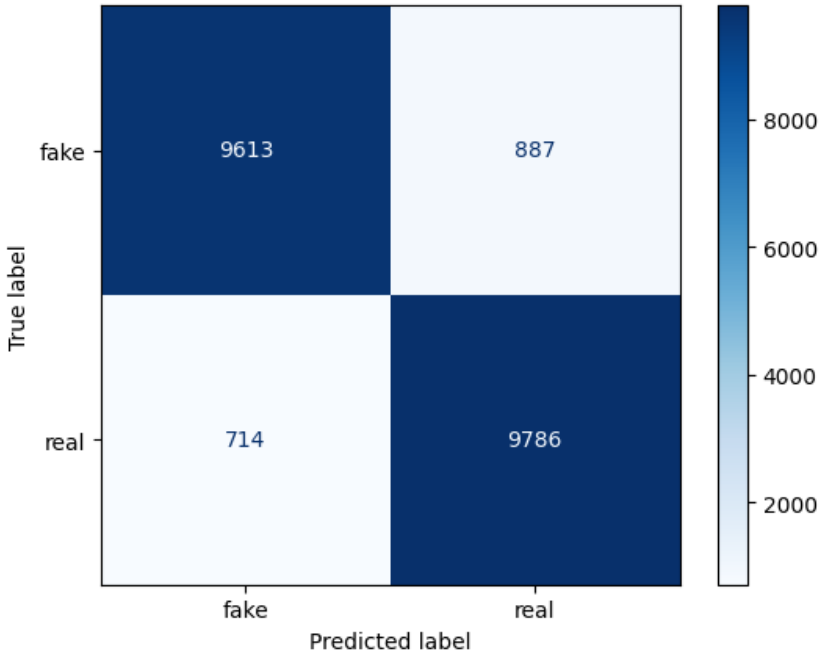

Figure 13: MobileNETV3-Large ArtiFact

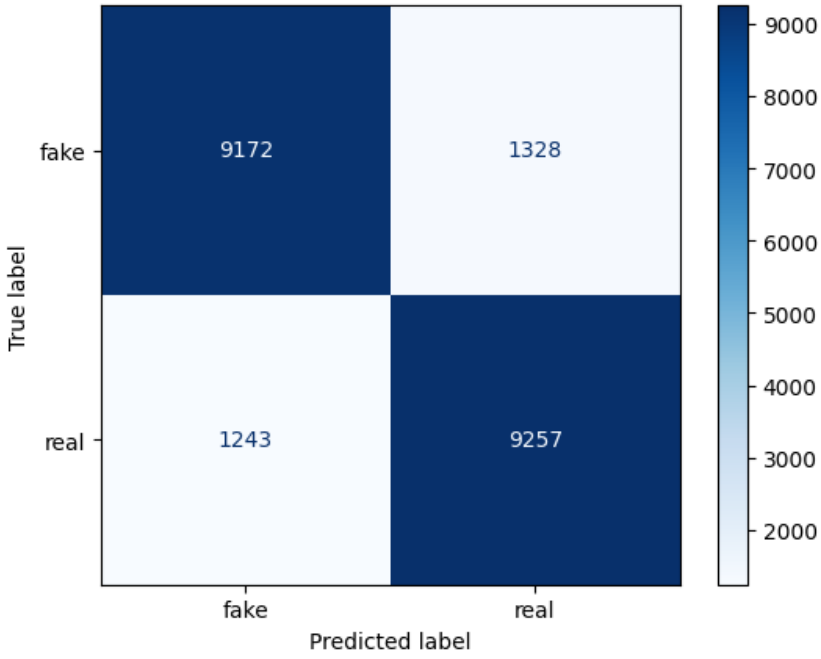

Figure 14: MobileNETV3-Small ArtiFact

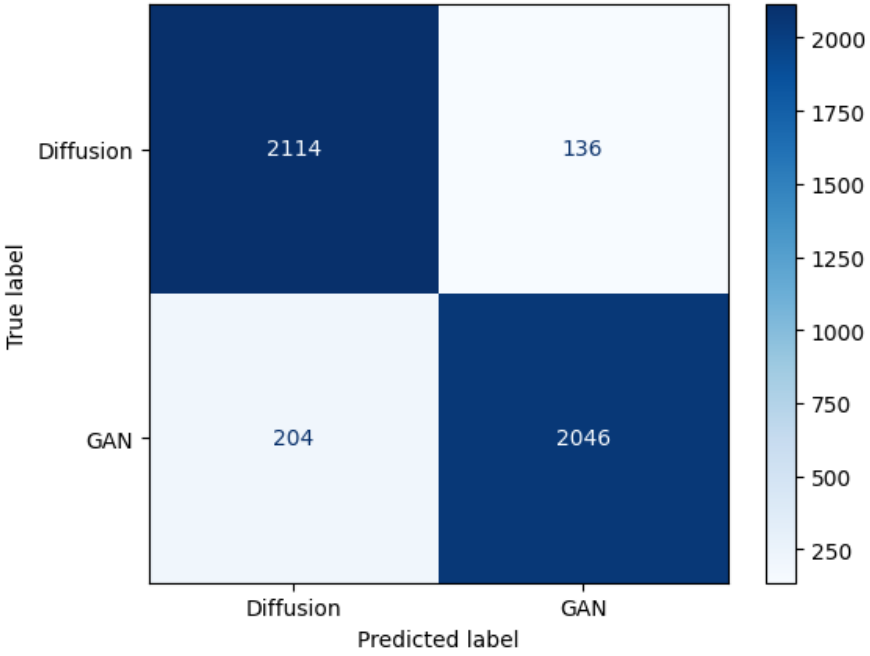

Figure 15: GAN vs Diffusion MobileNETV3-Large

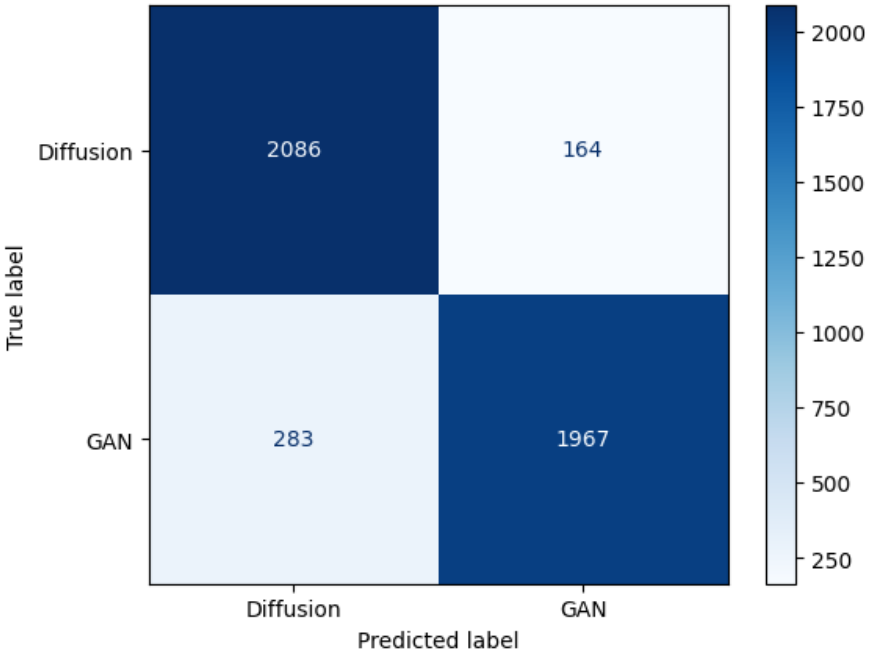

Figure 16: GAN vs Diffusion MobileNETV3-Small

Modell	Accuracy	Precision	Recall	F1-score
MobileNETV3-Large	**0.9214**	**0.94**	**0.92**	**0.92**
MobileNETV3-Small	0.9148	0.92	0.91	0.91

Table 6: Results Models GAN models

5.2.3 Model differentiation

The last level consists of two models and depends on the outcome of the GAN vs Diffusion level. One of the models differentiates between the GAN models and the other one between the Diffusion models. We use the MobileNetV3 models because they are the most efficient model on a mobile device. For the multi-class classification evaluation, the classification_report function is used which gives us an overview of all the metrics. Precision, recall, and F1-score are given per each category. We use the macro average for precision, recall, and F1-score, which means all the classes are weighted the same. We also produce the confusion matrices with the metrics.ConfusionMatrixDisplay function. The matrices are then plotted with matplotlib. This library is a Python library that can create visualizations that are static, animated, and interactive [53].

GAN Model Differentation Table 6 shows the results of the differentiation between the different GAN models. Precision, recall, and F1-score are the macro averages. The results can be seen in Table 6. MobileNETV3-Large performs better than MobileNETV3-Small in all the metrics. In Figure 17 we see that MobileNETV3-Large correctly predicts all the images generated by the models, BigGAN, StarGAN. Out of all the image generators images generated by StyleGAN2 are classified wrongly the most. 21.33% of wrongly classified StyleGAN2 images are classified in StyleGan1 so they seem to generate images that are similar to each other. This can also be seen in Figure 18. The images that are classified wrongly in StyleGAN2 are mostly classified as StyleGAN1 again. MobileNETV3-Small does not classify all the StyleGAN1 images correctly. Interestingly the falsely classified StyleGAN1 images are not classified as StyleGAN2 at all. So StyleGAN2 has a lot in common with StyleGAN1 but not the other way around. Another reason could be that the images generated by StyleGAN1 and StyleGAN2 can not be clearly distinguished so they are all classified as StyleGAN1. Figure 17 and Figure 18 use abbreviations for better readability, their full names can be found in Appendix B.

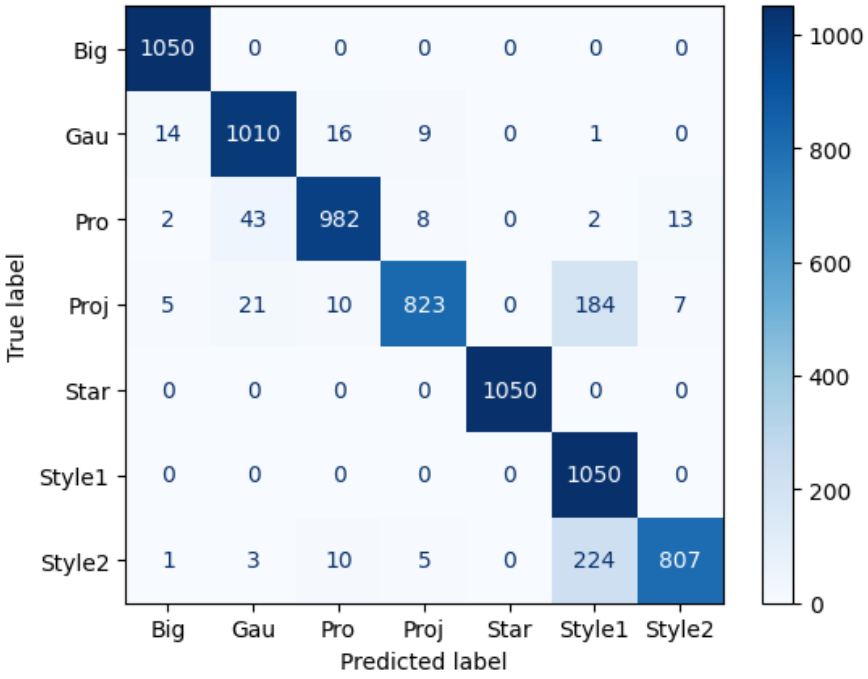

Figure 17: GAN Generators-Large

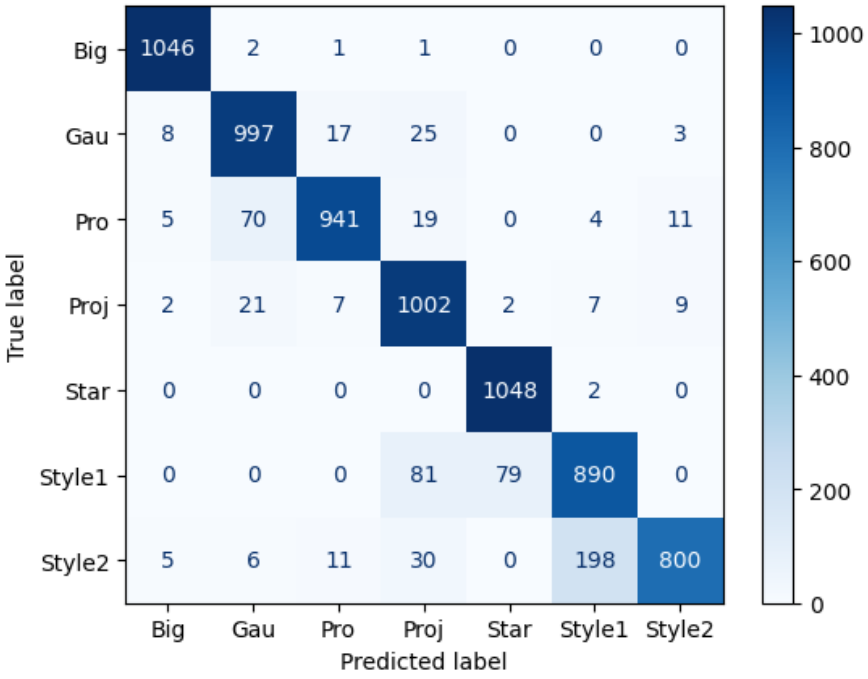

Figure 18: GAN Generators Small

Modell	Accuracy	Precision	Recall	F1-score
MobileNETV3-Large	**0.928**	**0.93**	**0.92**	**0.92**
MobileNETV3-Small	0.9209	0.92	0.9	0.91

Table 7: Results Models Diffusion Models

Diffusion Model Differentation As shown in Table 7 MobileNETV3-Large performs better than MobileNETV3-Small with an accuracy of 0.928. Both of the models achieve over 0.9 in all the metrics. The confusion matrices that can be seen in Figure 19 and Figure 20 show that the wrongly classified images from Latent Diffusion and Stable Diffusion are mostly classified wrong for each other. This does not necessarily stem from them generating images the most similar to each other but can come from Glide having fewer images in the training data. There are otherwise no abnormalities in the confusion matrices.

5.3 Evaluation and Comparison of the overall framework.

We evaluated the overall framework by building a loop which can be seen in Algorithm 2. This loop takes an image and puts it first through the AI vs real model. If the image is classified as fake it will be predicted in the GAN vs Diffusion model. If the image is predicted as Diffusion the image will be classified further into the Diffusion models. If it is predicted as GAN the classification continues between the different GAN models. At the end, we have two arrays. One with the real labels and one with the predicted labels. We then use the metrics.classifiation_report function to get the metrics for this result and plot a confusion matrix. MobileNETV3-Large performs better than MobileNETV3-Small with an accuracy of 0.81.

Stable Diffusion, StyleGAN1, ProjectedGAN, and GauGAN have the most realistic images from the model's perspective, images generated by these models are classified as real the most.

This can be seen in Figure 21 and Figure 22. StarGAN is nearly classified completely right from both of the models so images generated from StarGAN seem to have something that makes them easily identifiable. Figure 21 and Figure 22 use abbreviations for better readability, their full names can be found in Appendix B.

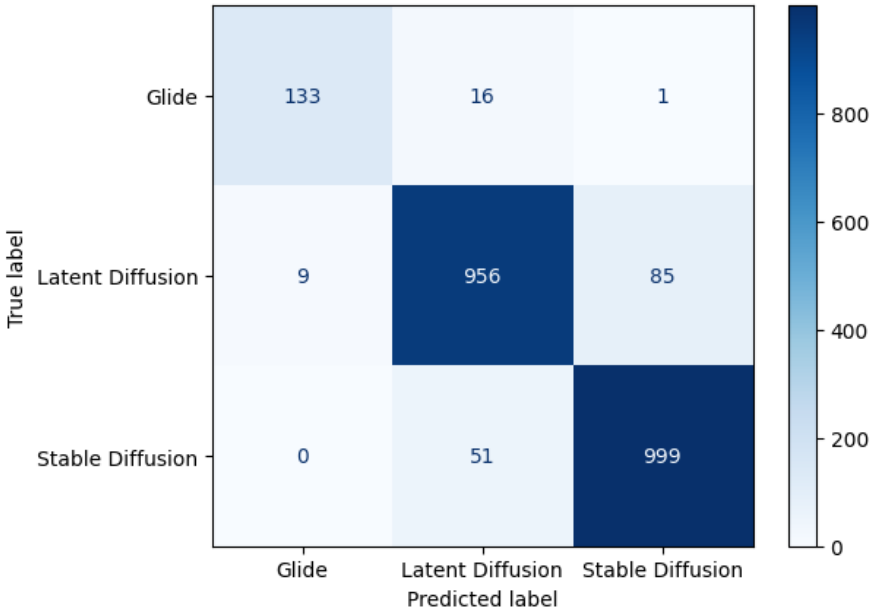

Figure 19: Confusion matrix Diffusion Models Large

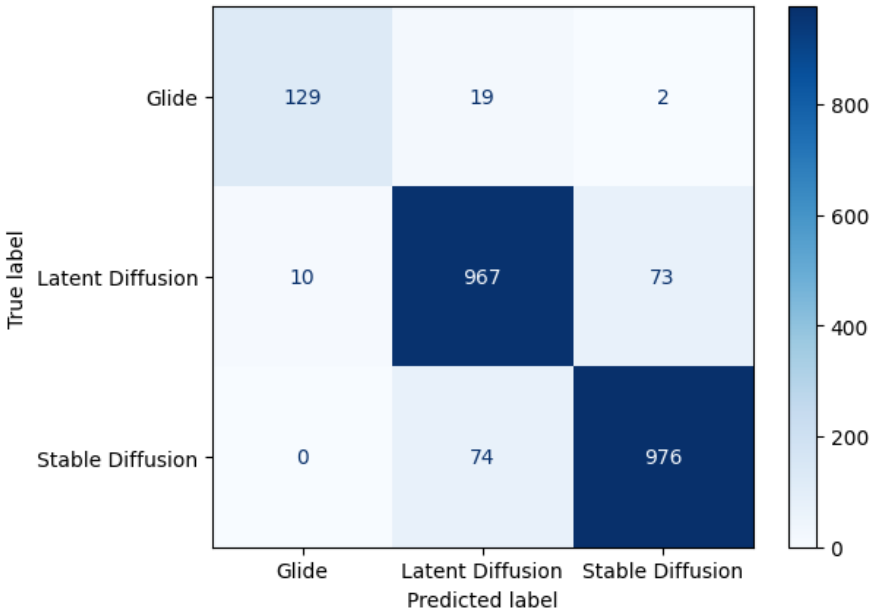

Figure 20: Confusion matrix Diffusion Models Small

Algorithm 2 The loop to evaluate the overall Framework

```
methodsTrue ← []
methodsPred ← []
pictures ← [list_of_pictures]
for all pic in pictures do
    methodsTrue ← add(method)
    testImage ← load_img(pic_path)
    testImageArray ← img_to_array(testImage)
    testImageArrayPreprocessed ← preprocess_input(testImageArray)
    testImageArrayDimension ← expand_dimensions(testImageArrayPreprocessed)
    result ← predictAIvsReal(testImageArrayDimension)
    if result=Real then
        methodsPred ← add(method_Predicted)
    else if result=Fake then
        result_vs ← predictGANvsDiffusion(testImageArrayDimension)
        if result_vs=Diffusion then
            result_Diff ← predictDiffusionModels(testImageArrayDimension)
            methodsPred ← add(method_Predicted)
        else if result_vs=GAN then
            result_GAN ← predictGANModels(testImageArrayDimension)
            methodsPred ← add(method_Predicted)
        end if
    end if
end for=0
```

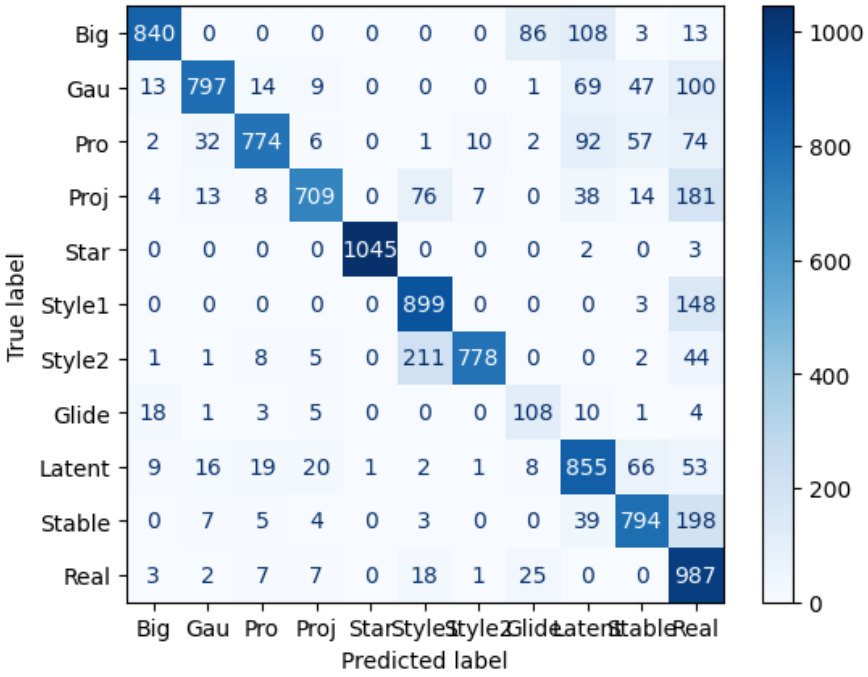

Figure 21: Confusionmatrix Overall Framework Large

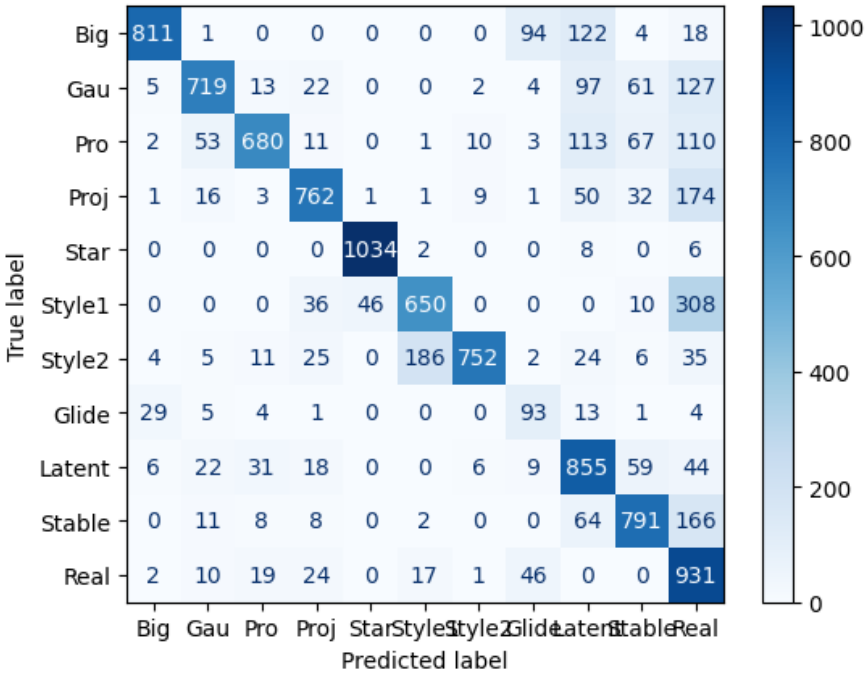

Figure 22: Confusionmatrix Overall Framework Small

Modell	**Accuracy**	**Precision**	**Recall**	**F1-score**
MobileNETV3-Large	**0.81**	**0.81**	**0.80**	**0.79**
MobileNETV3-Small	0.76	0.77	0.75	0.74

Table 8: Results Overall Framework

5.4 Limitations

Our biggest limitation is the limited computational power of the two laptops, we use to train the deep-learning models. This limited our possibility to train certain models. It also made perfecting the models more complicated because we could not use a bigger dataset for example.

The second limitation we had was our dataset. There was a limited number of datasets available, where it is clear which image is generated by which generator so for levels two and three of our framework we only had this dataset available. The limitation of this dataset is that the ratios are imbalanced. There are for example only 1000 images available from Glide. This also leads to the second limitation of the dataset which is that although the images are generated in the same categories they are not all generated with the exact same quotes. An example of this would be that there are two women faces one AI-generated one not, but the women look completely different. This makes it harder to evaluate if the patterns the model finds are related to the image being AI-generated or not. The model for example could also find that AI-generated women's faces have all blond hair because there are no real women with blond hair in the training data. If the AI-generated images would be generated based on a description of the real images this would be less of a concern.

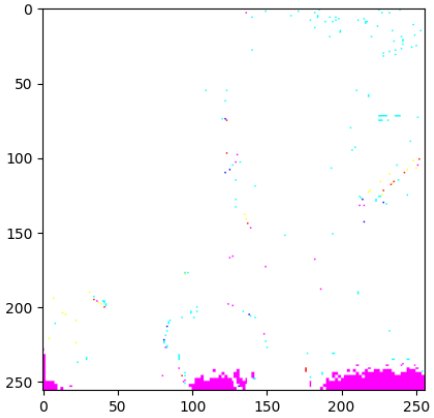

Figure 23: Before Scaling

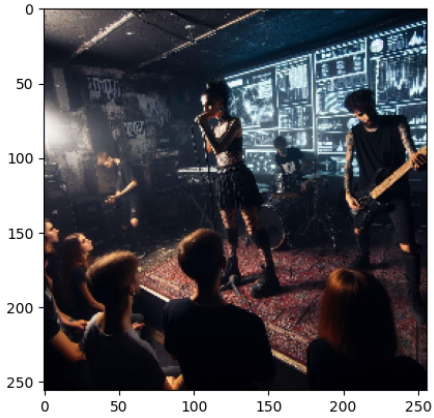

Figure 24: After Scaling

6 Model Decision Analysis

This chapter explores how the model decides if an image is AI-generated or not, for this SHAP [37] is used. SHAP is a Python package that helps to analyze the choice a machine-learning model makes [37]. As explained by Lundberg et al. [37] SHAP uses Shapley values from game theory, which are used to figure out how to distribute the cost to the people who consumed the service [52]. To explain a machine learning model the values are not people but the attributes in the model [52]. In the case of images, the attributes are pixels or areas in an image. These areas are analyzed on how they influence the result.

6.1 Implementation

We use SHAP on the models MobileNETV3-Large, DenseNet-121, and ResNet-101. These models are choosen because they are trained on the AIvsReal dataset. We use this dataset because the SHAP implementation takes place simultaneously with the training of the models on the ArtiFact dataset. The reason for this is limited time. A random sample of images from the AIvsReal dataset is analyzed and then manually expanded where patterns are expected.

At first, the images of the MobileNetV3-Large results are scaled to make them more interpretable. Figure 23 shows an example image before scaling and Figure 24 shows the same image after scaling. The preprocessing functions from ResNet-101 and DenseNet-121 also influence the look of the images but scaling them back influences the result of SHAP which can be

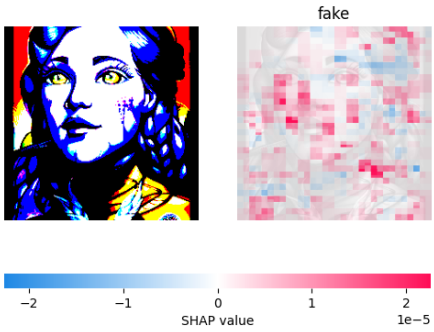

Figure 25: ResNet SHAP result preprocessed

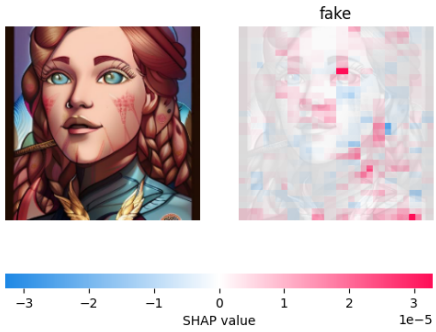

Figure 26: ResNet SHAP result unpreprocessed

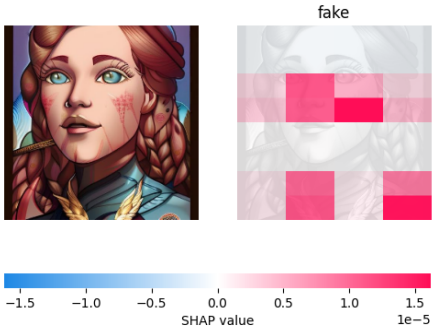

Figure 27: Example for SHAP results

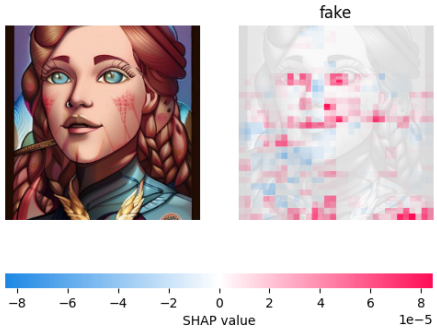

Figure 28: Example for SHAP results 7000

seen in the Figures 25 and 26. The preprocessed images are used because these are the ones that are used by the model to make its prediction. Multiple results were compared with the MobileNETV3-Large results but the rescaling did not influence the results so we use the rescaled version.

The masker is set to blur the background and the max evaluation value is set to 7000. Max evaluation influences how many evaluations are done. This value influences in how many areas the SHAP result is split. This can be seen when comparing Figure 27 and Figure 28. The first one uses a smaller value for max evaluation and the second one is 7000. In comparison to other values 7000 leads to the result that is the easiest for us to analyze and compare with each other. Figure 27 shows a result for a SHAP explanation on our image classification model AI vs Real. The label of this image is fake which means the pink sections indicate that the image is fake and the blue ones indicate that the image is real. The level of transparency shows how strongly the pixels influence the result and their interpretation can be seen under the

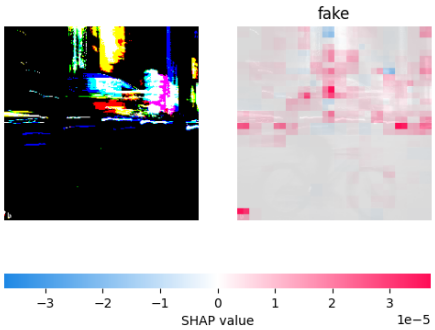

Figure 29: SHAP Result DenseNet-121 1

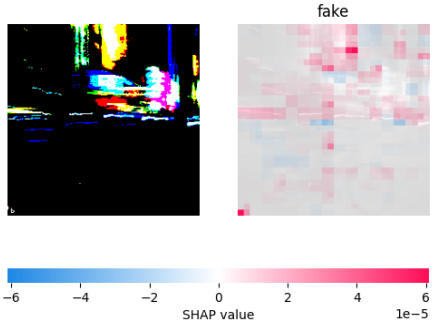

Figure 30: SHAP Result ResNet-101 1

Figure 31: SHAP Result MobileNETV3-Large 1

images.

6.2 SHAP Results

In this chapter, we analyze SHAP results based on five examples. We do this by comparing the results from different models on the same pictures.

6.2.1 SHAP Results Example 1

Figure 29, Figure 30 and Figure 31 show the results of SHAP on a fake image. DenseNET-121 and ResNET-101 classify the picture correctly and MobileNETV3-Large classifies the picture wrong. A sign that this image is fake is found in a small logo on the left bottom corner. Next to the logo DenseNet-121 and ResNet-101 find a lot of the pixels that influence the result in the regions of the image that show car lights or building lights.

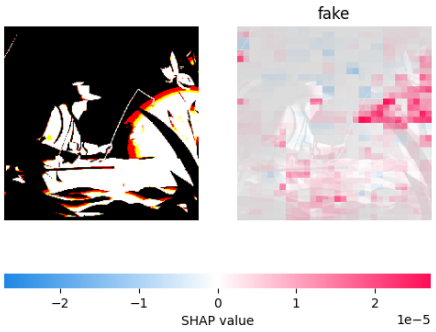

Figure 32: SHAP Result DenseNet-121 2

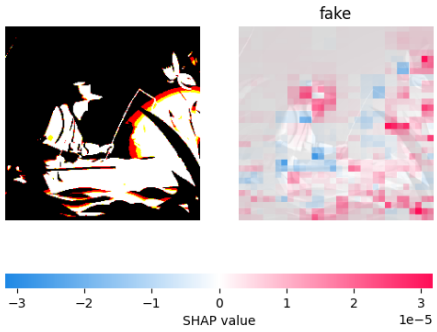

Figure 33: SHAP Result ResNet-101 2

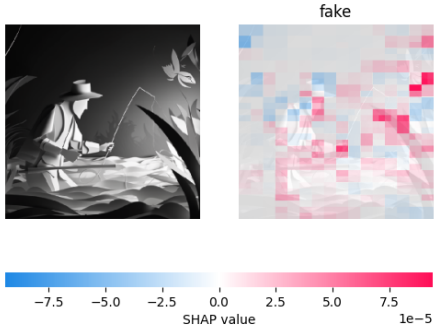

Figure 34: SHAP Result MobileNETV3-Large 2

6.2.2 SHAP Results Example 2

Figure 32, Figure 33 and Figure 34 show the result of SHAP on a fake image. All the models classified the image correctly. ResNet-101 and MobileNETV3-Large both see most of the classifiers in the figure and the grass and not in the darkness. DenseNet-121 focuses a lot on a region in the grass. Regions that indicate that the image is fake in ResNet-101 indicate that the image is real in MobileNETV3-Large like for example the bottom right corner. Another region where this difference appears is in one of the hands of the fisher. DenseNet-121 does not take these regions into consideration that much.

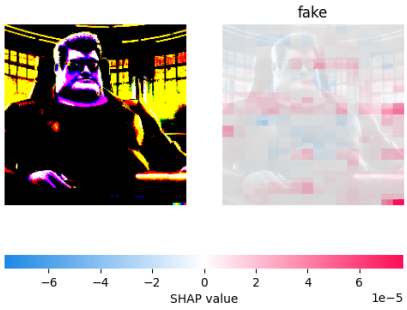

Figure 35: SHAP Result DenseNet-121 3

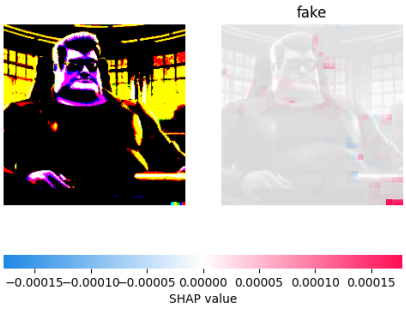

Figure 36: SHAP Result ResNet-101 3

Figure 37: SHAP Result MobileNETV3-Large 3

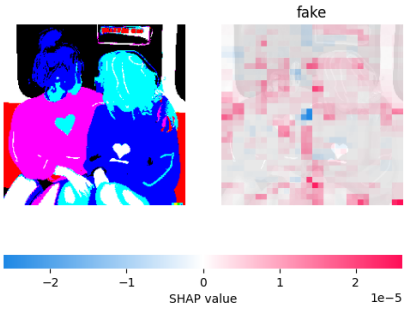

Figure 38: SHAP Result DenseNet-121 4

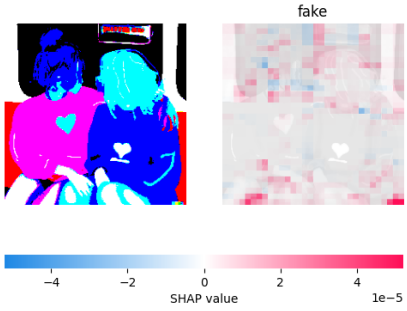

Figure 39: SHAP Result ResNet-101 4

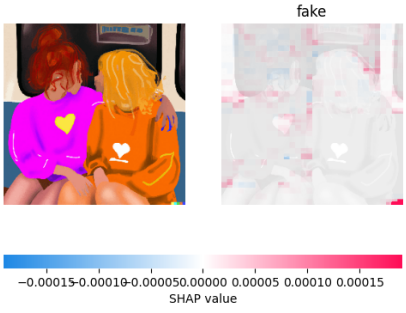

Figure 40: SHAP Result MobileNETV3-Large 4

6.2.3 SHAP Results Example 3

A sign that is used repeatedly to identify fake images are colour squares in the bottom right corner. This can be seen in Figures 35 to 40. These squares seem to be a good identifier of AI images in our dataset.

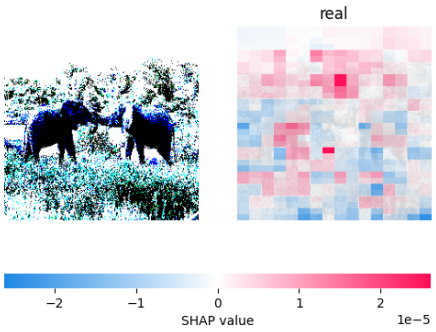

Figure 41: SHAP Result DenseNet-121 5

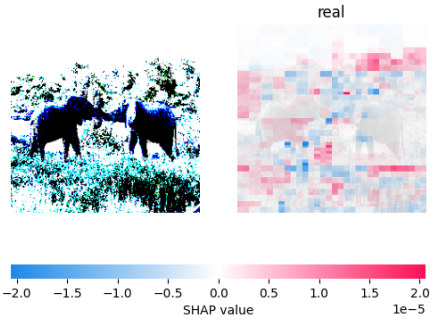

Figure 42: SHAP Result ResNet-101 5

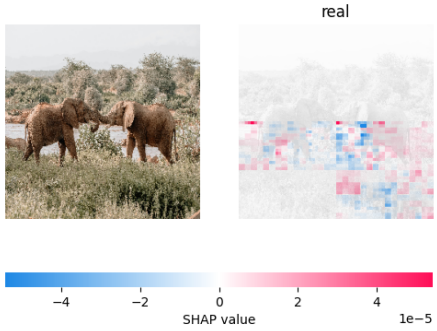

Figure 43: SHAP Result MobileNETV3-Large 5

6.2.4 SHAP Results Example 4

Figures 41 to 43 show a real image that is classified correctly by all the models. ResNet-101 considers the surroundings more than the elephants in the picture whereas MobileNETV3-Large focuses on the elephants and only a small part of the surroundings. DenseNet-121 takes nearly every region into account. The sky and trees indicate the most that the image is real. The grass and the elephant regions include the most part that speak for the image being fake. ResNet-101 analyzes similar regions but the regions that speak for and against the image being real are more scattered. There is also again the case that DenseNet-121 and ResNet-101 classify the same region exactly the opposite. DenseNet-121 classifies the region between the elephant's heads to speak for the image being real and ResNet-101 classifies it as showing that the image is fake.

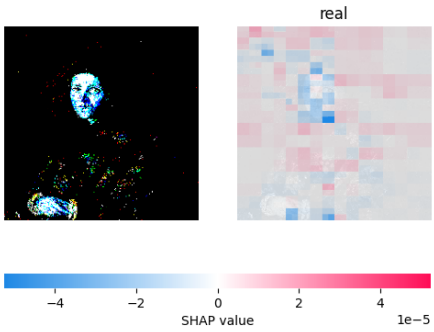

Figure 44: SHAP Result DenseNet-121 6

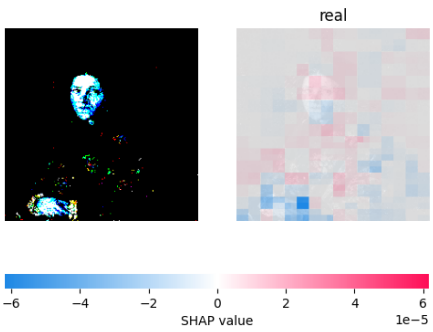

Figure 45: SHAP Result ResNet-101 6

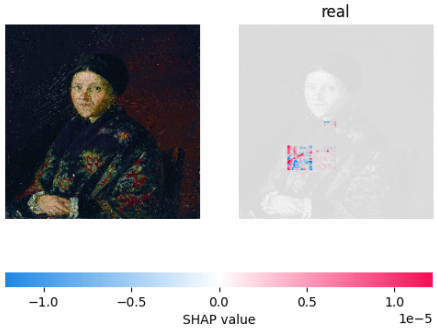

Figure 46: SHAP Result MobileNETV3-Large 6

6.2.5 SHAP Results Example 5

Figure 44 to Figure 46 show a real picture where all the models classified it correctly. MobileNETV3-Large makes its decision based on a small region of the woman in the painting. The regions DenseNet-121 and ResNet-101 look at are more scattered. ResNet-101 concentrates more on the woman and less on the surroundings. DenseNet-121 takes the surroundings also into consideration. DenseNet-121 finds a lot of classifiers that signal against the image being real in the face. ResNet-101 also takes the face into consideration. The top half indicates that the image is real and the bottom half against it.

By comparing all the results, we can see that the models take different regions into consideration in nearly all the images. Sometimes the same region is taken into account but it speaks for the opposite class. The only classifiers that is repeatedly taken as a sign that the image is fake are logos or colored squares in the bottom-left corner.

6.3 Important to take into consideration

Although the logos and squares are a good sign in our dataset this does not mean it is a sign for AI-generated images. The logo and the squares need to come from the training data so real images with these attributes need to exist. In our case, the real data is not the training data from the models but consists of real images where the copyright allowed them to be used in a dataset on Kaggle [61]. So this can be taken into consideration but it needs to be looked at in the context of the image. If an image from a website has this website's logo it should not be taken into consideration. If a person on Instagram posts a picture and claims it as their own with one of these signs it can be taken more into consideration but it still should not be the only attribute that leads to a decision.

7 Conclusion and further research

In this chapter, we summarize all the information from the thesis, by answering the research questions. Then we elaborate on how this research could be continued in the future.

7.1 Conclusion

The research question of this thesis is: How can we determine which model generated an image? This question is split into the sub-questions:

- How can we detect AI-generated images?
- What models generate images that are the hardest to identify?
- How does the algorithm decide whether an image is AI-generated?

To answer these questions we at first, analyzed how we can detect AI-generated images. We saw that humans have problems distinguishing AI-generated images from real ones with an image classification accuracy of 61 %. We also analyzed that deep learning models can be trained with the original image, the DFT of an image, the DCT of an image, and the inter-pixel correlation to identify AI-generated images. Another way to identify AI images is to use the 1D Power Spectrum of an image with machine learning models like SVM. We train deep-learning models with the original image to detect fake images. The best model achieved an accuracy of 0.9238 on one of our datasets.

To determine which model generated an image we build a 3-level framework that at first finds all the AI-generated images. Then split these into GAN and Diffusion models and finally split them into their generators. Each level consists of a deep learning model trained specifically for this use case. The overall model with the best overall accuracy of this framework achieved 0.81. The overall results show that StyleGAN 1, ProjectedGAN, and StableDiffusion generate the images that are classified the most as real.

The SHAP results show us that every model has different indicators for the differentiation between AI and Real images. Sometimes the same regions on an image even indicate different models the complete opposite of each other. There are some signs where the models agree like the color squares in the bottom corner but there is a high probability that is more dataset-dependent than AI-generated dependent.

7.2 Further research

Computational power limitations Further research on the topic of this thesis could be done by training ResNet-101 and DenseNet-121 with more computational power. The computational power limited our ability to optimize these two models and use them on the other framework levels.

Dataset In this thesis, we use two pre-existing datasets and only one to train the overall framework. It would improve our ability to find patterns in SHAP if we would have images that are as close to the real images as possible. Also, the datasets could be increased with more Diffusion models.

Specialized Datasets Another thing that could supplement the research conducted in this thesis is that the results of a one-categoric dataset could be compared to our results with the multicategoric dataset. So for example, the dataset only consists of human faces, to see if this influences the result. The result could be better because the model can concentrate more on the differences between AI-generated faces and real faces or worse because it does not generalize as well.

Robustness Testing To build on this research it would be good to test the results on their robustness. The results could be compared to other methods of AI detection for example predictions with the DFT of an image this way it could be said which method is the most robust.

References

[1] Zeeshan Ahmad, Zain ul Abidin Jaffri, Meng Chen, and Shudi Bao. Understanding gans: fundamentals, variants, training challenges, applications, and open problems. *Multimedia Tools and Applications*, pages 1–77, 2024.

[2] Safinah Ali, Daniella DiPaola, and Cynthia Breazeal. What are gans?: Introducing generative adversarial networks to middle school students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15472–15479, 2021.

[3] WASIN ALKISHRI and MAHMOOD AL-BAHRI. Deepfake image detection methods using discrete fourier transform analysis and convolutional neural network. *Journal of Jilin University (Engineering and Technology Edition)*, 42(2), 2023.

[4] Samah S. Baraheem and Tam V. Nguyen. Ai vs. ai: Can ai detect ai-generated images? *Journal of Imaging*, 9(10), 2023.

[5] Gwern Branwen. Danbooru2021: A large-scale crowdsourced and tagged anime image dataset. https://gwern.net/danbooru2021, 2021. Accessed: 2024-08-16.

[6] Peter Checkland and Sue Holwell. *Action Research*, pages 3–17. Springer US, Boston, MA, 2007.

[7] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. *CoRR*, abs/1912.01865, 2019.

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

[9] Linda Dickens and Karen Watkins. Action research: Rethinking lewin. *Management learning*, 30(2):127–140, 1999.

[10] Ricard Durall, Margret Keuper, Franz-Josef Pfreundt, and Janis Keuper. Unmasking deepfakes with simple features. *arXiv preprint arXiv:1911.00686*, 2019.

[11] Hugging Face. Diffusers: State-of-the-art diffusion models for image and audio generation in pytorch. https://github.com/huggingface/diffusers, 2024. Accessed: 2024-12-04.

[12] Python Software Foundation. os — miscellaneous operating system interfaces. https://docs.python.org/3/library/os.html, 2024. Accessed: 2024-10-08.

[13] Python Software Foundation. pathlib — object-oriented filesystem paths. https://docs.python.org/3/library/pathlib.html, 2024. Accessed: 2024-10-08.

[14] Python Software Foundation. random — generate pseudo-random numbers. https://docs.python.org/3/library/random.html, 2024. Accessed: 2024-10-08.

[15] Tegan George. What is action research? | definition & examples, January 2023. Accessed: 2024-07-02.

[16] Peter Goldsborough. A tour of tensorflow. *CoRR*, abs/1610.01178, 2016.

[17] Matthew Groh, Ziv Epstein, Nick Obradovich, Manuel Cebrian, and Iyad Rahwan. Human detection of machine-manipulated media. *Commun. ACM*, 64(10):40â€“47, September 2021.

[18] Luca Guarnera, Oliver Giudice, and Sebastiano Battiato. Mastering deepfake detection: a cutting-edge approach to distinguish gan and diffusion-model images. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2024.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[21] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019.

[22] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[23] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[24] Getty Images. Generative ki von getty images, unterstÃ¼tzt von nvidia., n.d. Accessed: 2024-07-16.

[25] Adobe Inc. Adobe firefly: die nÃ¤chste generation der generativen ki., 2024. Accessed: 2024-07-16.

[26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[27] Tero Karras, Samuli Laine, Timo Aila, and NVIDIA. Stylegan: A style-based generator architecture for generative adversarial networks. https://github.com/NVlabs/stylegan, 2019. Accessed: 2024-12-04.

[28] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila, and NVIDIA. Stylegan2: An improved version of stylegan for image synthesis. https://github.com/NVlabs/stylegan2, 2020. Accessed: 2024-12-04.

[29] Tero Karras and NVIDIA. Progressive growing of gans for improved quality, stability, and variation. https://github.com/tkarras/progressive$_g rowing_o f_g ans$, 2018. *Accessed* : $2024 - 12 - 04$.

[30] Kera. About keras, n.d. Accessed: 2024-07-20.

[31] Surya Widi Kusuma, Friska Natalia, Chang Seong Ko, and Sud Sudirman. Detection of ai-generated anime images using deep learning. *ICIC Express Letters, Part B: Applications*, 15(3):295 â€“ 301, 2024. Cited by: 0.

[32] NVidia Labs. Ffhq dataset, 2019. Accessed: 2024-11-11.

[33] NVIDIA Labs. Spade: Spatially-adaptive normalization. https://github.com/NVlabs/SPADE, 2021. Accessed: 2024-12-04.

[34] OSUâ€™S CENTER FOR TEACHING LEARNING. *ACTION RESEARCH*, n.d.

[35] Guifang Lin and Wei Shen. Research on convolutional neural network based on improved relu piecewise activation function. *Procedia Computer Science*, 131:977–984, 2018. Recent Advancement in Information and Communication Technology:.

[36] Zeyu Lu, Di Huang, LEI BAI, Jingjing Qu, Chengyue Wu, Xihui Liu, and Wanli Ouyang. Seeing is not always believing: Benchmarking human and model perception of ai-generated images. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 25435–25447. Curran Associates, Inc., 2023.

[37] Scott M. Lundberg and Su-In Lee. An introduction to explainable ai with shapley values. https://shap.readthedocs.io/en/latest/example$_n$*otebooks*/*overviews*/*An Accessed* : 2024 − 09 − 04.

[38] Jyoti Mann. Meta is using your instagram and facebook photos to train its ai models, May 2024. Accessed: 2024-07-02.

[39] Midjourne. Midjourney, n.d. Accessed: 2024-07-16.

[40] OpenAI. Glide: Text-to-image generation. https://github.com/openai/glide-text2im, 2021. Accessed: 2024-12-04.

[41] OpenAI. DallÂ·e 2, 2022. Accessed: 2024-07-02.

[42] OpenAI. DallÂ·e 3, n.d. Accessed: 2024-07-02.

[43] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[45] Andreea Pocol, Lesley Istead, Sherman Siu, Sabrina Mokhtari, and Sara Kodeiri. Seeing is no longer believing: A survey on the state of deepfakes, ai-generated humans, and other nonveridical media. In *Computer Graphics International Conference*, pages 427–440. Springer, 2023.

[46] Nihal Poredi, Deeraj Nagothu, and Yu Chen. Authenticating ai-generated social media images using frequency domain analysis. In *2024 IEEE 21st Consumer Communications Networking Conference (CCNC)*, pages 534–539, 2024.

[47] Danfeng Qin, Chas Leichner, Manolis Delakis, Marco Fornoni, Shixin Luo, Fan Yang, Weijun Wang, Colby Banbury, Chengxi Ye, Berkin Akin, Vaibhav Aggarwal, Tenghui Zhu, Daniele Moro, and Andrew Howard. Mobilenetv4 – universal models for the mobile ecosystem, 2024.

[48] Md Awsafur Rahman, Bishmoy Paul, Najibul Haque Sarker, Zaber Ibn Abdul Hakim, and Shaikh Anowarul Fattah. Artifact: A large-scale dataset with artificial and factual images for generalizable and robust synthetic image detection. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 2200–2204, 2023.

[49] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Niessner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[50] Shawn Shan, Wenxin Ding, Josephine Passananti, Stanley Wu, Haitao Zheng, and Ben Y Zhao. Nightshade: Prompt-specific poisoning attacks on text-to-image generative models. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 212–212. IEEE Computer Society, 2024.

[51] Bloomberg Technology Summit. Meta's cox on building ai into the product suite, May 2024. Accessed: 2024-07-02.

[52] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In Hal DaumÃ© III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9269–9278. PMLR, 13–18 Jul 2020.

[53] The Matplotlib Development Team. Matplotlib. https://matplotlib.org/stable/, 2024. Accessed: 2024-10-08.

[54] Unknown. DALLÂ·E by OpenAI. https://openai.com/index/dall-e/. [Accessed: September 18, 2024].

[55] Unknown. Diffus - AI Tools and Models. https://www.diffus.me/. [Accessed: September 18, 2024].

[56] Unknown. Pexels - Free Stock Photos. https://www.pexels.com/de-de/. [Accessed: September 18, 2024].

[57] Unknown. Unsplash - Beautiful Free Images Pictures. https://unsplash.com/de. [Accessed: September 18, 2024].

[58] Unknown. WikiArt - Visual Art Encyclopedia. https://www.wikiart.org/. [Accessed: September 18, 2024].

[59] Unknown. Keras. https://keras.io/api/applications/mobilenet/, 2015.

[60] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A. Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[61] Tristan Zhang. Ai-generated images vs real images. https://www.kaggle.com/datasets/tristanzhang32/ai-generated-images-vs-real-images, 2023. Accessed: 2024-08-16.

[62] Nan Zhong, Yiran Xu, Sheng Li, Zhenxing Qian, and Xinpeng Zhang. Patchcraft: Exploring texture patch for efficient ai-generated image detection, 2024.

A Appendix -Citations of the Datasets

The AIvsReal dataset is from Tristan Zhang taken from kaggle from the url: https://www.kaggle.com/datasets/tristanzhang32/ai-generated-images-vs-real-images/data The following MIT license text is taken from

https://www.mit.edu/ amini/LICENSE.md which is the link that is connected to the dataset on kaggle.

> *"Released under MIT License*
> *Copyright (c) 2013 Mark Otto.*
> *Copyright (c) 2017 Andrew Fong.*
> *Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:*
> *The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.*
> *THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE."*

The Artifact dataset consists of multiple datasets combined with different licenses taken from: https://www.kaggle.com/datasets/awsaf49/artifact-dataset

The FFHQ dataset is made available under the Creative Commons BY-NC-SA 4.0 license by NVIDIA Corporation [32]. This allows us to use this dataset by naming changes we made. The biggest change we made is that we only use a subset of it. We also have to give credit by citing their paper [26]. The last point is that we have to distribute any derivative works under the same license.

The license text in [32] also states that each image has an individual license, which can be found in the metadata. We are allowed to use all of the images in our usecase and the links to the metadata can be found in the following license text taken out of [32]:

> *The individual images were published in Flickr by their respective authors under either Creative Commons BY 2.0, Creative Commons BY-NC 2.0, Public Domain Mark 1.0, Public Domain CC0 1.0, or U.S. Government Works license. All of these licenses allow free use, redistribution, and adaptation for non-commercial purposes. However, some of them require giving appropriate credit to the original author, as well as indicating any changes that were made to the images. The license and original author of each image are indicated in the metadata.*
>
> https://creativecommons.org/licenses/by/2.0/
>
> https://creativecommons.org/licenses/by-nc/2.0/
>
> https://creativecommons.org/publicdomain/mark/1.0/
>
> https://creativecommons.org/publicdomain/zero/1.0/
>
> http://www.usa.gov/copyright.shtml

The ImageNET dataset is made available under the Non-commercial license which allows us to use this dataset. The full dataset can be found at: https://www.image-net.org/download.php

The AFHQ dataset is made available under the Creative Commons Attribution-NonCommercial 4.0 International Public License, which allows us to use this dataset for non-commercial purposes. The dataset was created in the context of the paper [7]. This dataset can be found at:

https://github.com/clovaai/stargan-v2?tab=readme-ov-file. The only change we made to this dataset is that we use a subset of it. The whole license text can be found at: https://github.com/clovaai/stargan-v2?tab=License-1-ov-filereadme

BigGan, Glide, Latent Diffusion, Projected GAN and StarGAN are made available under the MIT license [48]. This license allows us to use it as long as we include the copyright notice [40]. For Glide this notice is taken from https://github.com/openai/glide-text2im?tab=MIT-1-ov-filereadme and is

Copyright (c) 2021 OpenAI Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute,

sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

For Latent diffusion, this copyright notice is taken from

https://github.com/compvis/latent-diffusion?tab=MIT-1-ov-filereadme

and says: *MIT License Copyright (c) 2022 Machine Vision and Learning Group, LMU Munich Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.*

For ProjectedGAN this notice is taken from

https://github.com/autonomousvision/projected-gan?tab=MIT-1-ov-filereadme

and says: *Copyright (c) 2021 autonomousvision Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.*

For StarGAN this notice is taken from

https://github.com/yunjey/StarGAN?tab=MIT-1-ov-filereadme and says: *Copyright (c) 2017 Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.*

For BigGan this notice is taken from:

https://github.com/ajbrock/BigGAN-PyTorch/blob/master/LICENSE

Copyright (c) 2019 Andy Brock Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish,

distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

GauGAN uses the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International license which allows us to use it as long as it's not used for commercial purposes which we do not do [33].

ProGAN uses the Attribution-NonCommercial 4.0 International license which allows us to use it as long as it's not for a commercial purpose which we do not do [29]. Also credit has to be given: Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved. A link to the license has to be provided: https://github.com/tkarras/progressive$_g rowing_o f_g ans?tab = License-1 - ov - file$

and the changes we made have to be shown which are that we only use a subset of the dataset.

Stable Diffusion uses the Apache-2.0 License which allows our usage as long as we add a copy of the license [11]. The license can be found here

https://github.com/huggingface/diffusers?tab=Apache-2.0-1-ov-filereadme (added by link to save space).

StyleGan1 uses the Attribution-NonCommercial 4.0 International which allows us to use it for NonCommercial purposes as long as credit is given Copyright (c) 2019, NVIDIA CORPORATION [27]. All rights reserved. Also a link to the license has to be provided:

https://github.com/NVlabs/stylegan?tab=License-1-ov-filereadme

and the changes we made have to be shown which are that we only use a subset of the dataset. The authors of our dataset only named that StyleGAN 1 uses a Creative commons Public License which is why we use the name of the license that goes more into detail here.

StyleGan2 uses the Nividia Source Code License which allows us to use it for Non-Commercial purposes [28]. We also have to add a copy of the license: https://github.com/NVlabs/stylegan2?tab=License-1-ov-filereadme (added by link to save space). Should we ever make the dataset public we will do so under the same license for this subset so this point is also followed. The License states at the beginning: *Copyright (c) 2019, NVIDIA Corporation. All rights reserved.*

B Appendix -Table of the Abbreviations in the Figures

Abbreviation	**Full Name**
Big	BigGAN
Gau	GauGAN
Pro	ProGAN
Proj	ProjectedGAN
Star	StarGAN
Style1	StyleGAN1
Style2	StyleGAN2
Latent	Latent Diffusion
Stable	Stable Diffusion