Bachelor's Thesis Computing Science

Radboud University Nijmegen

Detection of AI-Made Artworks With One-Class Classification Algorithms

---

Author:
Alperen Özkurt
s1037565

First supervisor/assessor:
Twan van Laarhoven

Second assessor:
Tom Heskes

July 2, 2023

Abstract

This research investigates the utility of several one-class classification algorithms in distinguishing human-made paintings from AI-generated ones. This investigation questions the general use of one-class-based novelty detection algorithms while aiming for an analysis based on the metadata captured during this research. The research utilizes several classification algorithms, namely One-Class SVM, Isolation Forest, Local Outlier Factor, One-Class Neural Network, and Robust Convolutional Autoencoder. This study was conducted due to the insufficient amount of literature available on the varied applications of such algorithms and the researcher's own fascination with artworks. For the experiments, we use a collection of paintings gathered from WikiArt and OpenArt websites with the extra data related to these paintings. Then, we propose several detectors to make the distinguishing between real and fake paintings. The best-performing detector extracts the features of the images with a residual neural network. Following this extraction, it standardizes these extracted features and reconstructs them afterward by an autoencoder we introduce. Finally, the one-class classifier, built into the detector, gives a final decision on the class of the data given as input. While feature extraction makes this research computationally feasible, input processing with an autoencoder increases the final accuracy of the model. The majority of the test results show a high rate of false positives and false negatives. Although the AUC scores indicate that these algorithms perform better than random guessing, they demonstrate a significantly lower accuracy compared to a multi-class classifier. Based on the accuracy and the occurrences in the experiments, the paper provides a metadata analysis. In conclusion, the overall status of one-class classification is uncertain, and it is evident that multi-class classifiers excel in this particular task.

Contents

Chapter 1

Introduction

Text-to-image AI models are getting more popular day by day. In the middle of 2022, with the release of Midjourney, DALL-E, and Stable Diffusion, the popularity of these models increased rapidly, see Figure 1.1. America and Europe are interested in DALL-E while Asian countries are mostly interested in Midjourney. And Stable Diffusion is the most popular one in very few countries, such as Japan and South Korea. Interestingly Stable Diffusion is an open-source model, while the others are unfortunately not. And yet Stable Diffusion is the least favorite among these models. Nevertheless, some people embraced it and published their own modified versions of it[1].

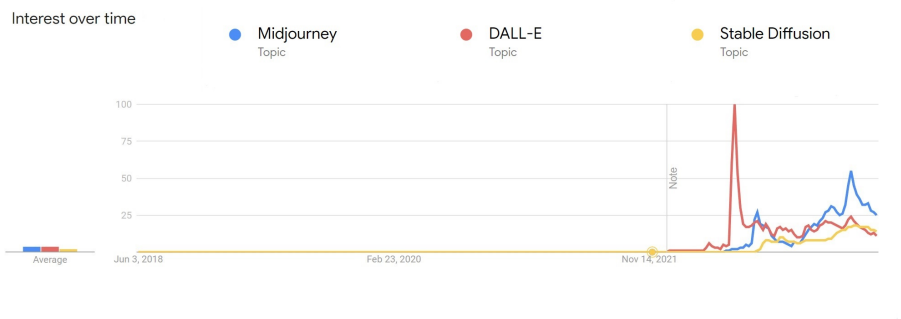


Figure 1.1: Search trends of text-to-image models in the world by Google[2]

The images created by these models are not perfect, in the sense that they are distinguishable to some degree with our very human eyes, whether these images are artificially created or not. On this matter, computers are better than humans to make this differentiation. Binary classifiers built by

[1] `https://huggingface.co/models?other=stable-diffusion`

[2] Interest over time: Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. A score of 0 means there was not enough data for this term. (Defined by Google)

other scientists give 86% accuracy scores[1]. The same type of classifiers are also built by us. And the results are nearly consistent with each other with a score of 93%. Confidently, we can call these models quite successful.

However, the focus of this paper is not building on top of these binary models. We want to emphasize the very first time we have seen the images created by text-to-image models. Naturally, we found them easily differentiable by their essence. But how did we make such a differentiation? We all have a variety of ideas of what an image should look like in our experiences. We had never seen an AI-made image once. We had this pile of memory of images and nothing much else. We compared what we see in front of our eyes with the ones we saw in the past. And at the end, we concluded they could not be normal images. These artificial images were outside of the definition of the images we know. Or they had certain anomalies that can be easily detected. For example, poor representation of a human face, distortions, sharpness, or blurry parts of an image are ways to detect these abnormalities in human face imitation. Figure 1.2 shows an example of such cases. Briefly, the detection of an abnormality by knowing only one side like the aforementioned is called novelty detection.

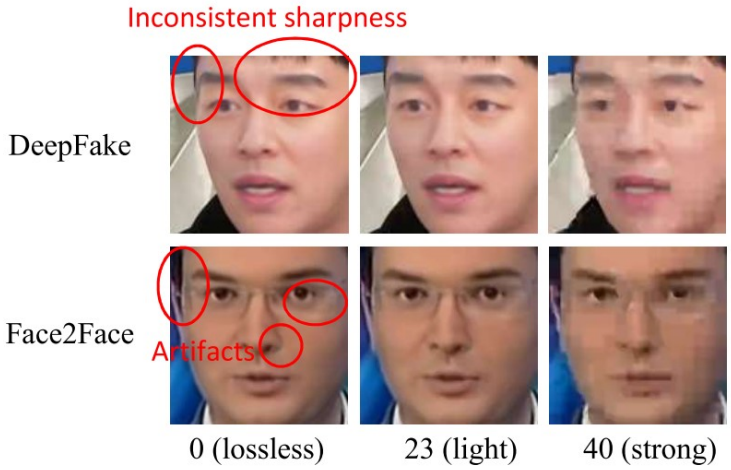


Figure 1.2: Video compression results of forged media using H.264 codec. DeepFake shows inconsistent sharpness along the borderline of the facial region whereas Face2Face contains some artifacts. The compression laundries the manipulation traces from the data. [2]

Although the terms abnormality detection and anomaly detection are generally used as the synonym of outlier detection, in this paper they are used as the synonym of novelty detection. In this detection, we use one-class classification (OCC) methods where the model is trained with one type of class only. Applications of anomaly detection are numerous and are mainly around cyber security, e.g., fraud detection, intrusion detection, diagnosing

patients, and applying law... There are a variety of techniques that can be applied in order to achieve this detection. Commonly used methods are support vector machines (SVMs), density-based techniques like local outlier factor (LOF) and isolation forest (IF), and neural networks. A decent ratio of high positive or strong negative utility would lead to success with these algorithms. However, the possibility of convergence to the training set picked is a dangerous potential and is very likely to be happening. Therefore, parameter tuning, proper data set selection, and neural network structure (if applicable) are vital for healthy abnormality detection outcomes. As a result of the training with one class only, the definition of normal is constructed. Then, anything outside of the norms designated by the algorithm is counted as an anomaly. By the specifics of how novelty detection is defined, we find it close to human nature. Therefore, we found it interesting. Moreover, we want to see if this problem-solving technique can be used in any case or not. Eventually, this research questions the generality of the one-class classification algorithms while creating an automated Turing test for text-to-image models.

The reason we chose specifically the paintings is our interest in art. And not just us but the people were quite interested in this matter with the rise of NFTs in 2021[3] and the aforementioned text-to-image models release. With the public release of ChatGPT[4], we believe the spotlight on the prompt-based models is now on ChatGPT. This research started earlier than the rising popularity of this chatbot model, so we kept our focus specifically on this issue with paintings.

So, ultimately, the main research question of this paper is can we distinguish artificially-made paintings that are created by various text-to-image models from human-made paintings with one class classification algorithms? (RQ-1)

In the following section, the background knowledge is supported for the people unaware of one-class classification algorithms, residual networks, and text-to-image models and how they work. Then, we introduce the images we use in this research. Afterward, we show the detectors we built to answer RQ-1 and the environments that these detectors run in. Later on, the results that are produced by the detectors we built are introduced. Following that, the analysis of the metadata is shown in detail. Finally, the limitations and future work are discussed.

[3] `https://trends.google.com/trends/explore?date=today5-y&q=nft,/g/11rsc2xsp1&hl=en`

[4] `https://chat.openai.com/`

Chapter 2

Preliminaries

This chapter explains one-class classification algorithms, residual neural networks, and text-to-image models that are related to this research.

2.1 One-class classification (OCC) algorithms

One-class classification algorithms are designed for scenarios where data from a single class is available for training a model. The primary objective of these algorithms is to construct a model capable of differentiating instances that belong to the target class from those that do not. In essence, their purpose is to identify and flag outliers or anomalies within a data set.

By introducing different algorithms, we also want to answer another question which is 'can we spot a performance division between differentiating algorithms? (RQ-2)'. The first three methods, OC-SVM, IF, and LOF, are imported from the scikit-learn (sklearn) Python library[3]. And the rest of the two methods are imported from the implementation of other researchers. There is also the Elliptic Envelope method from sklearn, which did not cope with our coding setup due to internal errors and unappreciated computation time.

2.1.1 One class support vector machines (OC-SVM)

Support vector machines (SVMs)[4] are commonly used for generalizing input data. For non-linear classification, SVM applies the kernel trick first. The kernel-trick is a method where non-linear data is projected onto a high dimensional space. The trick aims to make the classification linearly dividable by a plane. After these high-dimensional feature spaces are created, SVMs start defining a set of hyper-planes in the multidimensional space to make the classifications. Optimally, the gap between these hyper-planes should be as high as possible in order to minimize generalization because generalization leads to error. Data points that are closest to a hyperplane are

called support vectors. And based on the kernel function chosen, boundaries are decided for the final classification.

With OC-SVM, firstly, we apply the same kernel-trick. Then, we separate all data points from the origin in the feature space using a hyper-plane. Points below the hyper-plane and closer to the origin are called outliers. In this research, the implementation of scikit-learn is used, which uses the version of Schölkopf et al.[5].

2.1.2 Isolation forest (IF)

Isolation Forest[6] is a variation of the decision tree algorithm. Outliers get isolated by a randomly selected feature from the given feature space and then randomly selected a split value between the minimum and maximum values of that feature. More specifically, via parameter-controlled random subsampling, binary trees are created. Afterward, for each point in the data set, the average path length required for the decision is calculated. This is also known as the anomaly score of a data point. The lower the anomaly score, the algorithm marks those points as outliers.

2.1.3 Local outlier factor (LOF)

Local outlier factor algorithm[7] also uses the term anomaly score. This score measures how much the density of a particular sample deviates from its neighboring samples. The concept of locality is crucial here, as the anomaly score relies on the degree of isolation of a sample within its immediate neighborhood. To estimate this locality, the LOF algorithm considers the distances to the k-nearest neighbors and uses them to gauge the local density. By comparing the local density of a sample to the densities of its neighbors, we can identify samples that exhibit significantly lower density values. Such samples are considered outliers.

2.1.4 Robust convolutional autoencoder (RCAE)

RCAE[8] is a model that initially aims to strengthen principal component analysis (PCA). PCA is a powerful tool to perform dimension reduction and feature extraction but it is sensitive to outliers. Moreover, it has a linearity assumption, which might cause terrible results in nonlinear variations. By introducing robust PCA, they allow some input to be randomly distorted, which might create testing errors in the end. Then, by establishing an autoencoder that absorbs the created noise, they aim to make learning possible in a nonlinear subspace. However, such noise injection and absorption, and the following autoencoder method are designed to be used in anomaly detection. Therefore, while its state for general usage is questionable, it solves the problem for our task. Moreover, RCAE is introduced to be superior to the other one-class algorithms, which is highlighted in Figure 2.1.

METHOD	AUC
OC-SVM/SVDD	52.5 ± 1.3
KDE	51.5 ± 1.6
IF	53.37 ± 2.9
AnoGAN	-
DCAE	79.1 ± 3.0
SOFT-BOUND. DEEP SVDD	67.53 ± 4.7
ONE-CLASS. DEEP SVDD	67.08 ± 4.7
OC-NN	63.53 ± 2.5
RCAE $\lambda = 0$	87.45 ± 3.1
RCAE	$\mathbf{87.39 \pm 2.7}$

Figure 2.1: Average AUCs in % with StdDevs (over 10 seeds) per method on GTSRB stop signs[9] with adversarial attacks. [8]

2.1.5 One class neural network (OC-NN)

OC-NN is a model for unsupervised anomaly detection introduced by Chalapathy et al.[10], which is inspired by his previous work with RCAE. It combines OC-SVM and autoencoders by creating its own loss function. OC-NN refines features of objects to aim anomaly detection with the objective it defines. But by doing that they give up on reaching global optima. When convergence occurs, the line between what is normal and abnormal is decided, and it classifies the data points.

Even though it is introduced later than RCAE, it does not come up with better results on the same tasks. And when it comes to comparing it with other algorithms like OC-SVM and IF, depending on the task, it either outperforms these algorithms or stays somewhere near them, see Figure 2.1.

2.2 ResNet

2.2.1 What is ResNet?

ResNet is basically the abbreviation of residual networks, which are made up of Residual Blocks[11]. Residual blocks allow the network to directly propagate the original input to the deeper layers of the network, which is called skipping connections with identity mapping. These blocks enable efficient training in very deep networks. Figure 2.2 represents a simple residual building block.

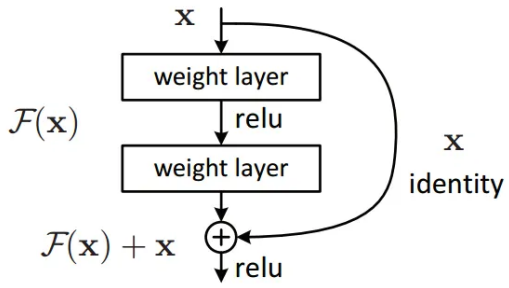


Figure 2.2: Residual learning: a building block with identity function x and deeper residual function $\mathcal{F}$. [12]

Most of the popular residual networks combine these blocks with convolutional layers to extract features from images or any other multidimensional feature space. ResNet[12] is the most famous example of this category and is commonly used in object detection, image segmentation, and image classification. These tasks can succeed after adding 1 to 3 additional layers to the end (after the extraction of features) and training with an appropriate data set afterward. Consisting of 1000 classes and 14 million images, ImageNet[1] is the most famous example used for multi-class image classification. Such a classification would suit our research problem here if it was not multi-class in its sense. Therefore, we only used it for feature extraction for the artworks collected. See Figure 2.3 for architectural details of different ResNet builds.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2.3: ResNet architectures for ImageNet. Building blocks are shown in brackets with the number of blocks stacked. Downsampling is performed by conv3 1, conv4 1, and conv5 1 with a stride of 2. [12]

[1] `https://www.image-net.org/`

2.2.2 The most appropriate version of ResNet for this project

By expanding the number of residual blocks, it is possible to have a much deeper feature extraction. One of the famous neural network Python libraries TensorFlow 2[13] implements ResNet-50, ResNet-101, and ResNet-152. The numbers dictate the number of layers in the network. Using ResNets or convolutional neural networks(CNNs) solely for image classification shows that as the amount of layers increases the better accuracy we get in the end, see Table 2.1.

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 2.1: Error rates (%, 10-crop testing) on ImageNet validation. Other networks are included for comparison purposes. ResNet-34 is option A. ResNet-50/101/152 is option B which only uses projections for increasing dimensions.[12]

Obviously, ResNet is not a one-class classifier. But with the combination of the algorithms mentioned earlier, it can be a part of the process. Maybe because of this combination, digging into deep features does not provide a performance rise specifically on this task. On top of that, without staying neutral, a performance decline can be observed in some cases, like our task, see Table 6.2 for exact numbers in the result chapter.

2.2.3 Other networks for feature extraction

TensorFlow 2 provides other famous networks as well, such as Xception[14], EfficientNet[15], and VGG[16] as deep CNNs. And also DenseNet[17] which is a residual neural network like ResNet. The outcome of these models highly depends on the task. All of the individual papers of these models compared themselves to the other models in the past. Some are even superior to ResNet. Hence, we wanted to use those models for the feature extraction as well. However, in the end, ResNet-50 is picked in this research for the feature

extraction task. Because the final results achieved by ResNet-50 surpass the other models on the one-class classification problem on artworks. In the results chapter, the outcomes of these different models are introduced, see Table 6.3.

2.3 Text-to-Image models

This section introduces the different text-to-image models in our test data set. Although the mechanisms of these models are not directly relevant to our research, we still want to elucidate them for anyone with an interest in this matter. Conducting this preliminary research enabled us to discover the practicality of autoencoders, which we ultimately employ to develop our detectors.

2.3.1 Stable-Diffusion

Stable Diffusion[18] is one of the pioneers of text-to-image models. It is open-source and widely used and modified by the community. It is one of the first well-known models to introduce diffusion models. Diffusion models are a type of generative model designed to generate data that resembles the training data they were trained on. The core principle of diffusion models involves introducing Gaussian noise to the training data, progressively deteriorating its quality, and then learning to reverse this degradation process to recover the original data. Once trained, the diffusion model can generate new data by applying the learned denoising process to randomly sampled noise. The reverse degradation process for recovery is called reverse diffusion. Stable diffusion combines this reverse diffusion idea with a text transformer. Specifically, another model converts prompts to tokens. Then each token is converted to a 768-value vector called an embedding. By using a text transformer on this embedding, the final noise becomes ready for the reverse diffusion model. However, this process is not complete. Stable Diffusion is known for being faster and arguably more effective among diffusion models because of its latent structure, 48 times smaller than high dimensional image space thanks to variational autoencoding.

In each creation of an image, a random tensor is created in the latent space. The diffusion model takes the refined text prompt and the image created in the latent space and predicts the noise. The image in the latent space is redefined by subtracting the predicted noise from itself. This last process repeats itself for some iterations. In the end, the latent image gets decoded by the variational encoder and we got to receive the final output. See Figure 2.4 for a better understanding.

The insights from the variational autoencoder and the Stable Diffusion inspired this research to build our own autoencoder, which is introduced in further sections of this paper.

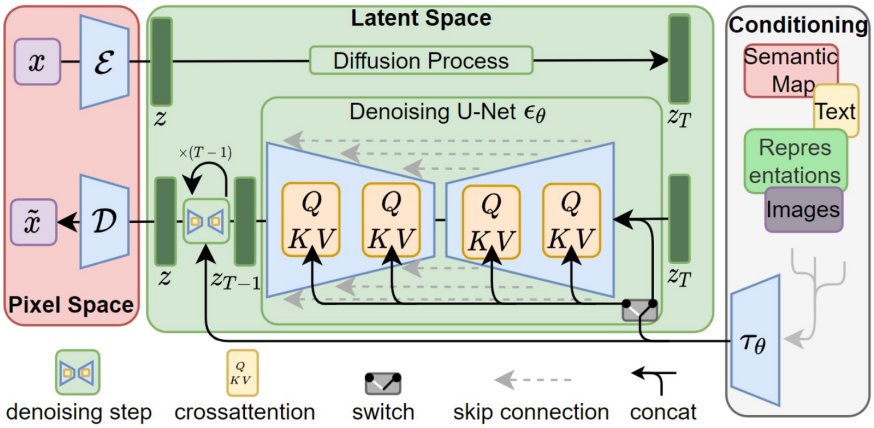


Figure 2.4: Stable Diffusion generates an image from a prompt: A random image x is encoded and transferred to latent space, while from the given prompt a text embedding is created. After this random image is diffused, the U-Net autoencoder that mixes embedding sequences(cross-attention) rebuilds the image. The output image $\tilde{x}$ is ready after the iterative denoising step and final decoding of the denoised image into pixel space. [18]

Based on the data it trained, there are multiple versions of Stable Diffusion. But LAION[19] is the provider of all these training data. Various versions of Stable Diffusion are apparent in our test data set as well as modified versions of it.

2.3.2 OpenAI's DALL-E

DALL-E[20] is another example of the success of diffusion models. As most of the terms are explained with Stable Diffusion, we can get straight with how DALL-E 2 works.

The process starts by using the CLIP text encoder. CLIP trained with the WebImageText data set consisting of 400 million images and their captions where the goal was capturing the relation between pixels and the natural language. Images and corresponding captions get encoded to convert the image description into a representation in a specific space. Next, the diffusion prior takes this CLIP text encoding and maps it to a corresponding CLIP image encoding. Lastly, the modified GLIDE generation model utilizes reverse diffusion to map from the representation space to the image space and that is our final output. See Figure 2.5 for a better understanding.

CLIP is a vital part of this whole DALL-E 2 idea. Creating an enormous correlation map of words and images is fascinating. Training CLIP in different eras would be worth-seeing research. Other than CLIP, any improvement by computational means that autoencoders can supply is mentioned

and noted by the designers of DALL-E but not applied. Yet it is still one of the fastest and one of the most widely used models outside.

2.3.3 Midjourney

The developers of Midjourney[2] do not provide any public information on the mechanism they are using. Hence, the technical part of Midjourney is nothing more than this. But assuming the variation system that Midjourney has, the generative model structure can be similar to DALL-E. Moreover, thinking about the quality of the images that Midjourney produces, they are clean from noise and have high resolution in general. These are all the intuitions that we believe this model is similar to DALL-E. On the other hand, based on our personal experiences, the correlation between objects and prompts feels off from time to time which makes us think they might be using a different technique from OpenAI's CLIP.

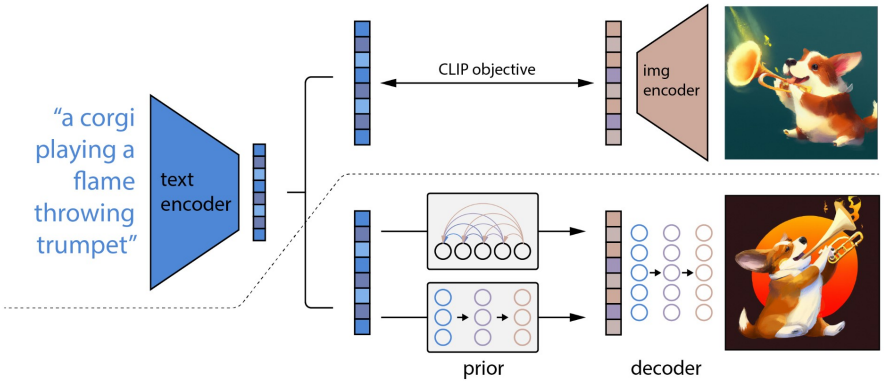


Figure 2.5: A high-level overview of unCLIP. Above the dotted line presents the training progress of CLIP which combines texts and images in the same space. Below the dotted line shows text-to-image generation from the text embedding or diffusion prior, which is given as input to the final diffusion decoder in order to create the ultimate image. [20]

[2] https://www.midjourney.com/

Chapter 3

Related Work

In 2021, Seliya et al. produced a covering literature research[21] on the applications of one-class classification. The paper discusses the issues that may arise when using OCC methods for novelty detection on large amounts of data. We can list some of these threats as noisy data, feature selection, and data reduction. The issues with OCC in big data clusters can also happen in any data set and harm OCC's results. Therefore, we implement an outlier removal algorithm for addressing noise reduction in our training set. Moreover, to our detectors, we introduce applying PCA and a reconstructing autoencoder in order to overcome feature selection and data reduction issues.

The aforementioned paper is an arguably recent literature review. As the releases of text-to-image models explained in our research are near mid-2022, it is not possible to find literature related to our research question (RQ-1) in that paper. Therefore, we seek other resources, such as DE-FAKE[1], deep learning-based OCC for fake faces[22], and the OC-NN paper[10]. These three research papers are closest to our topic and inspired us.

DE-FAKE paper builds binary classifiers on fake images in general. Not just that, the metadata that comes with the images is used in a hybrid approach to improve final results. Eventually, we implement a binary classifier to make a comparison with our results, as mentioned earlier. Additionally, even though we do not build any detectors with a hybrid approach, we discuss it in the discussion chapter later on.

Unlike DE-FAKE, the second paper uses an OCC algorithm but specifically focuses on human faces. Therefore, we realize there is a knowledge gap in the specific area where we conduct our research.

The OC-NN paper not only offers multiple practical settings for constructing our detectors but also presents comprehensive data on the efficacy of OCC algorithms applied to diverse tasks. It highlights the complexity of our objective in contrast to others, primarily due to the high-resolution images in our data sets.

Chapter 4

Data

This chapter of this paper introduces the images we collected, sources of them, and the metadata that comes along with them.

4.1 Data set of human-generated artworks

We gathered all of our human-made images from WikiArt[1]. WikiArt is a digital visual art encyclopedia that consists of more than 250000 artworks by 3000 artists of various styles. In this research, only about 81000 of them are used which is the same fraction used in artGAN once[23]. More specifically, there are 27 different styles, 1119 artists, and 81444 unique artworks in total.

The information available from this portion of the WikiArt data set is, per each artwork, the artist's name, the name of the painting, the style that is used on the painting, the width of the resolution, and the height of the resolution. Except for the names of the paintings, the rest of the data is used for further analysis. Further explanation on excluding names of the paintings is given in the discussion chapter.

The choice of using this set is appropriate and accurate as we know all the images are human-made. Moreover, we believe the essence of human-made artworks is well-represented in this collection. It contains plenty of artwork from a variety of artists with a reasonable amount of different painting subjects(portrait, genre painting, landscape...) and painting styles(realism, abstract...). Famous artworks such as The Starry Night by Van Gogh, Da Vinci's Mona Lisa, Gustav Klimt's The Kiss, and many more can be found for instance. This data set can be reduced or extended by further researchers because the motives behind our choices are subjective at points as explained.

[1] `https://www.wikiart.org/`

4.2 Data set of AI-generated artworks

When it comes to AI-generated images, we collected all of them from OpenArt[2]. The website collects AI-created images from various channels as well as the metadata belonging to these images. One of the reasons we choose this website is it contains more metadata variety in comparison to other websites like Lexica[3]. However, this OpenArt website has one downside, it does not specify prompt tags. For instance, Midjourney splits the prompt input into several tags, such as subject, medium, environment, lightning, color, mood, and composition[4]. Such prompt tagging would allow us to prepare a more detailed metadata analysis at the end. Another worth mentioning part about this website is OpenArt filters out the majority of the images created with NSFW prompts. Therefore, nude paintings are excluded from this website, which is completely fine as the WikiArt version that we use also does not contain any nude paintings.

With the scraper we implemented, we scrape the images that use the style of the most occurring 50 artists from our WikiArt set. We specifically do that scraping by searching "by" + "name of the artist" and exporting all the images and the metadata that comes along with them. Ultimately, we collected 260240 images between 6 May 2023 and 7 May 2023, which reduces to 130082 images after excluding duplicates. From now on, the name OpenArt set refers to the duplicates removed version of it. Future researchers can change their AI-made image collection as long as they keep the style of the famous painters being used in the artificially created images they picked.

Figure 4.1: Artworks from collected data. The first row is human-made images from Albrecht Dürer, Pablo Picasso, Claude Monet, Gustave Doré, and Vincent van Gogh respectively. The second row is AI-made images with the same artist order using their styles.

[2] https://openart.ai
[3] https://lexica.art/
[4] https://docs.midjourney.com/docs/prompts

4.3 Metadata

In this section, we introduce different types of metadata we collected that came with the images in our data set. While we explain these various metadata one by one, we include statistical distribution for some.

Artist: For the human-made artworks, the painter's name is used. For the AI-made images, the style of the artist used in the creation of the image fills this attribute.

Style: Style here is used in terms of visual arts. Therefore, it is about the visual aspects of a work of art that link it to other artworks created by the same artist, during the same time period, in the same place, cultural movement, or archaeological civilization. All human-made images have this attribute filled as WikiArt data have all images classified. However, no AI-made image has any value at all due to the lack of tags on the OpenArt website.

Model: For the AI-made images, the name of the text-to-image model created the image. It can be either different versions of Stable Diffusion, DALL-E, or Midjourney according to the OpenArt website. This attribute is a NaN value for the human-made ones. See the Figure 4.2 for detailed distribution of the models.

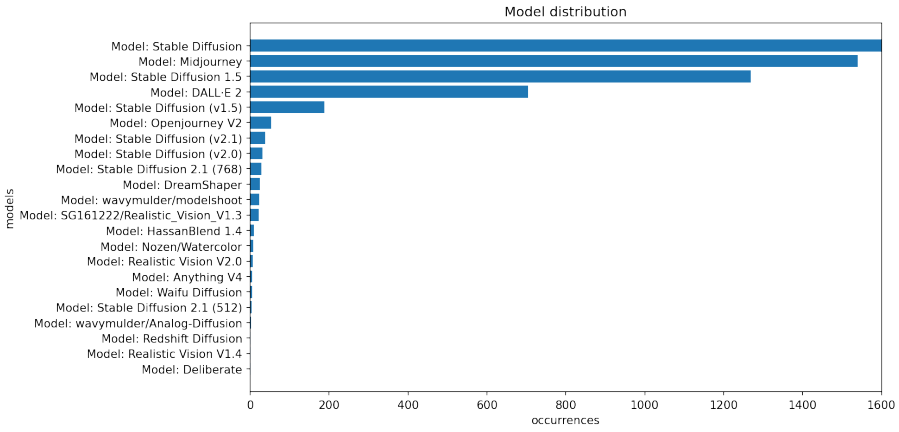


Figure 4.2: Distribution of the models in the OpenArt data set. There are 126109 images created by 'Model: Stable Diffusion', which is the 96.95% of the OpenArt set. All the models that do not mention Stable Diffusion, Midjourney, or DALL-E are the modified versions of Stable Diffusion that can be found on the Hugging Face website.

Prompt: Sentence or sentences used to describe what we want from the AI model to draw. When it comes to human-made images, this attribute stays empty/NaN. See the Figure 4.3 for the most used 30 words and for the words per prompt distribution.

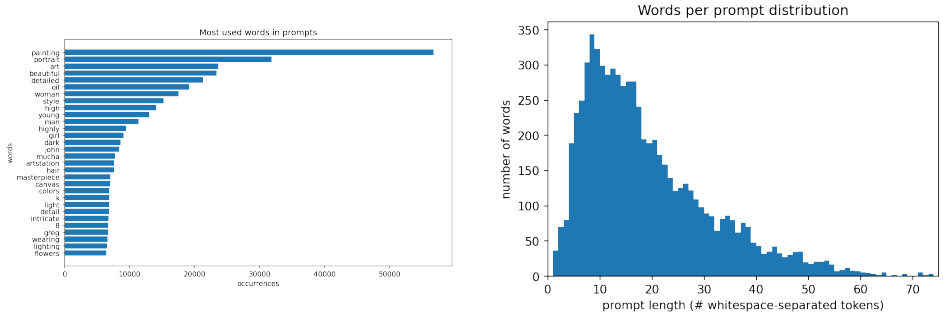


Figure 4.3: The left graph shows the most used 30 words in the prompts with a bar plot. Statistics exclude the stop words in English defined by NLTK[24]. The right one shows the distribution of the number of words used per prompt with a bar plot. The maximum number of words that can be interpreted in a prompt is around 75 for all models. Statistics include the stop words aforementioned.

Negative prompt: Sentence or sentences used to describe what we do not want from the AI model to draw. This is an optional parameter for the OpenArt set and NaN value for the WikiArt set. See the Figure 4.4 for the most used 30 words and for the words per prompt distribution.

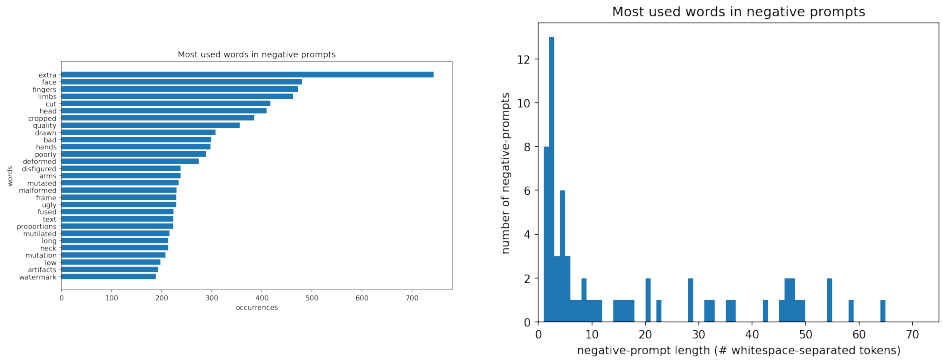


Figure 4.4: The left graph highlights the most used 30 words in the negative prompts with a bar plot. Statistics exclude the stop words in English defined by NLTK[24]. The right one shows the distribution of the number of words used per negative prompt with a bar plot. The maximum number of words that can be interpreted in a prompt is around 75 for all models. Statistics include the stop words.

Guidance scale: A parameter that manifests how much the image generation process follows the prompt. As the number goes higher, the more the model tries to follow the prompt given. There can be trade-offs in the final quality of the image generated. This parameter has the tendency to the default value of 7 in Stable Diffusion[18]. Not all the images created by AI models have this attribute, see Figure 4.5 for statistical distribution. For the majority of the images in the OpenArt set, this attribute is NaN (128286 images, 98.62% of the set), while it is also NaN for all the images in the WikiArt set.

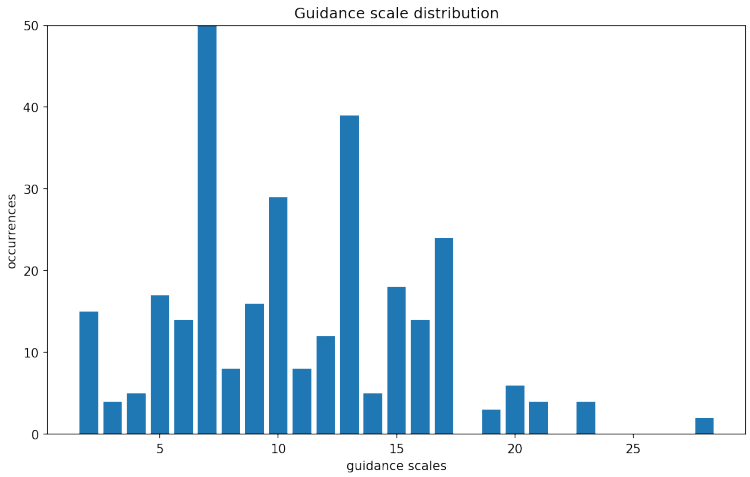


Figure 4.5: Distribution of the guidance scale parameter in the OpenArt data set. There are 1531 images created by the guidance scale 7, which is the 1.18% of the set.

Sampler: These models generate a completely random image first. Predicted noise is subtracted from the image repeatedly to get a clean image at the end. Samplers do the denoising part. They affect the final result and they are entirely optional. For some images in the AI-artworks, this attribute is NaN, while it is also NaN for all the images in the human-made artworks. Table 4.1 shows the samplers in the OpenArt data set.

	#	%
DDIM	83	0.06
DPM Solver++	1659	1.3
Euler A	35	0.03
Default/Unknown	128305	98.63

Table 4.1: Distribution of the samplers in the OpenArt data set.

Steps: The attribute refers to the number of denoising steps applied while creating an image by a text-to-image model which is explained in Figure 2.4 earlier. Some AI-made images have this attribute filled while no human-made image has any value at all.

Seed: The number is used to initialize the generation of the image created. All AI-made images have this attribute filled while no human-made image has any value at all.

Width: The vertical aspect of the resolution of the image created. All images have this attribute filled.

Height: The horizontal aspect of the resolution of the image created. All artworks have this attribute filled.

Resolution scale: The simplified fraction of the 'width' and 'height' attributes, also known as the resolution scale. All images have this attribute filled. Table 4.2 highlights the most occurring resolutions across the data we collected.

	1	2	3	4	5
WikiArt set	3 x 4 (462)	1 x 1 (420)	4 x 3 (328)	5 x 4 (200)	4 x 5 (162)
OpenArt set	1 x 1 (59698)	2 x 3 (14722)	8 x 11 (7886)	3 x 2 (5796)	2 x 1 (4708)

Table 4.2: Top 5 most apparent resolution scales with their amount in parentheses from the WikiArt set and the OpenArt set. There are 156 unique resolution scales for OpenArt, while WikiArt has 40039 unique resolution scales.

Chapter 5

Methods

This chapter introduces the detectors we built in order to answer our research questions. We also explain the experiment environment, in which these detectors run the tests.

5.1 Detectors

This section introduces the methods that we use to build our multiple different detectors. All of these detectors are built the same in their core. They train on a given one-class-only training set, then they predict if the objects of a test set are the same class as the training set or not. In our specific case, they train on human-made only images and then they try to predict the classes of images from a mixed test set. Therefore, these detectors play a key role to answer our first question, which is can we distinguish artificially-made paintings that are created by various text-to-image models from human-made paintings with one-class classification algorithms (RQ-1)? Then by building and testing these diverse detectors, we aim to get an answer for our second research question, which is can we spot a performance division between differentiating algorithms (RQ-2)?

The first type of detectors are called standard detectors, in the sense that they do not contain any additional pre-processing steps like PCA or reconstruction of input by autoencoders. Methods that are considered default/standard steps in this research are feature extraction, standardization of the data, outlier removal from the training set, and the usage of one-class classifiers. There are 3 different detectors under this category. The only difference is the one-class classification algorithm used at the end.

The second type of detectors are called PCA detectors. All of these detectors only add PCA on top of the standard detectors. This one also has 3 different detectors under this category. Again, the only difference is the one-class classification algorithm used at the end.

The third type of detectors are called reconstructing detectors. All of

these detectors only add reconstruction of the data with an autoencoder on top of the standard detectors. Just like the previous two, it comes with 3 different detectors, the only difference is the one-class classification algorithm used at the final step.

And the last type of detectors are called neural network decoders. These detectors are quite different from our other detectors and they essentially use neural networks in their core. There are two different detectors under this category. One uses RCAE and the other uses OC-NN.

5.1.1 Standard detectors

Firstly, all the images get their features extracted into a 2048-dimensional array via ResNet-50. Afterward, extracted features of given training get normalized by `StandardScaler()`[1]. Following that, the same transformation is applied by the standard scaler gets to be used in the given test set as well. By applying standardization, we aim to reshape the skewed output of the ResNet-50. Then, we assume there is a 2.5% contamination in our training set. Therefore, we get rid of them with an algorithm. We use the interquartile range to detect outliers in every dimension. For each dimension, if an element's value is outside of the interquartile range, its outlier score increases. After every dimension is covered, we remove the top-scoring 2.5% elements, from the training set. See Algorithm 1 for the pseudo-code. After these outliers are removed, we train OC-SVM, IF, or LOF with the decontaminated training set, which means there are 3 different standard detectors. Finally, we introduce the normalized test set to print results. For a better understanding, the architecture is provided with Figure 5.1.

Algorithm 1 Algorithm for the removal of the outliers.

```
function OutlierRemoval(X : 2D Array)
    scores ← list with all zeros, size of |X_0|
    for each i in |X_0| do
        Q1 ← lower quarter percentile value of X_i
        Q3 ← upper quarter percentile value of X_i
        IQR ← Q1 − Q3
        upper ← indices with the values stays above Q3 + 1.5IQR from X_i
        lower ← indices with the values stays below Q1 − 1.5IQR from X_i
        for each index in upper do
            scores_index ← scores_index + 1
        end for
        for each index in lower do
            scores_index ← scores_index + 1
        end for
    end for
    scores ← scores but only the top-scoring 2.5% with their indices alone
    X ←X.drop(scores)
    return X
end function
```

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

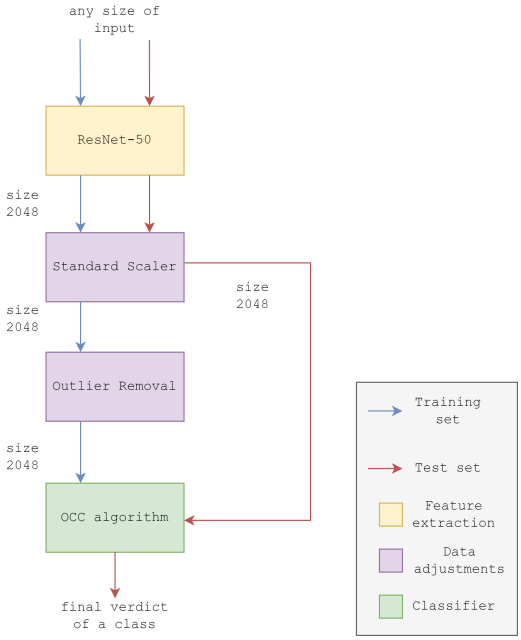


Figure 5.1: Architecture of the standard detectors. ResNet-50 extracts the features of the given input. Then, the standard scaler normalizes the output. Training set gets sanitized by Algorithm 1. An OCC algorithm, IF, OC-SVM, or LOF, trains on this cleansed set. Finally, the OCC algorithm receives the test set and outputs predictions.

5.1.2 PCA detectors

The aforementioned PCA can help one-class algorithms in two ways we foresee. The first one is it can help to discard noise or irrelevant features in the feature space by reducing dimensions. In terms of variance in the data, the top contributing attributes stay. And the second one is we can use the reconstruction error that might occur in PCA for our anomaly detection. The outputs with higher reconstruction errors have the potential of being anomalies. Therefore, we add PCA for our detectors.

These detectors also apply the same procedures at the beginning. ResNet-50 extracts the features, both sets are normalized, and outliers get removed from the training set. Then, just before putting everything into our one-class algorithms, we apply PCA to both the training and test set. We adjusted PCA, so it reduces 2048 dimensions to 512 dimensions. More components can lead to useless PCA. And fewer components can end up with information loss. Therefore, we introduce only 512 components. However, we want to emphasize that this parameter is not optimized, not ultimate. Hence, better results can be achieved with other numbers.

Then, we train OC-SVM, IF, or LOF with the reshaped and polished

training set, which means there are 3 different PCA detectors. Finally, we introduce the test set to print results. The architecture of the PCA detectors is shown in Figure 5.2.

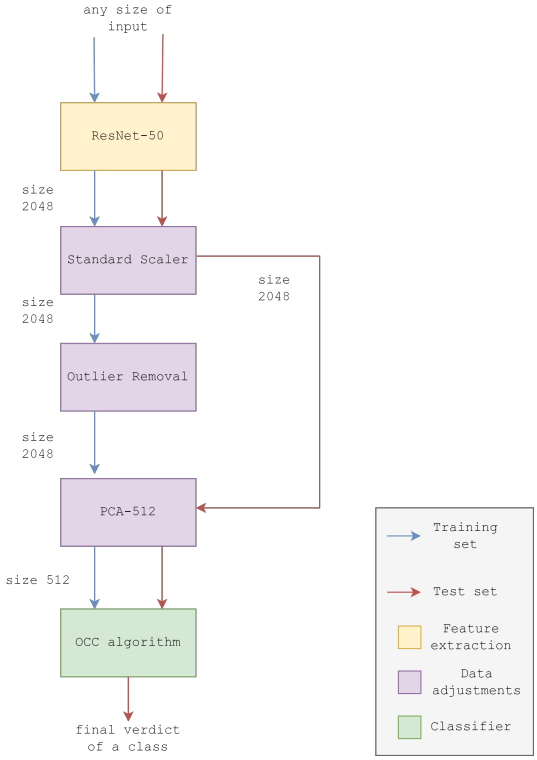


Figure 5.2: Architecture of PCA detectors. ResNet-50 extracts the features of the given input. Then, the standard scaler normalizes the output. Training set gets sanitized by Algorithm 1. After that PCA reduces the dimensionality of the cleansed training and the test set. Now, an OCC algorithm, IF, OC-SVM, or LOF, trains on the reconstructed training set. Finally, the OCC algorithm receives the modified test set and outputs predictions.

5.1.3 Reconstructing detectors

Inspiration from the Stable Diffusion, RCAE, and the reconstruction idea of PCA leads to our third detector. We introduce a simple and straightforward autoencoder in order to re-explore our inputs. Architecture is provided in Figure 5.3. Few remarks here, convolutional layers are not used in this autoencoder as we already use them with ResNet-50. Additionally, the amount of hidden layers is found to be sufficient as adding more or excluding some reduces the accuracy of the whole detector.

Just like other detectors, this one also applies the same procedures initially. ResNet-50 extracts the features, both sets are normalized, and outliers

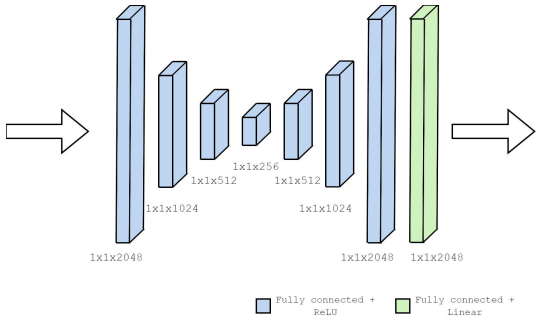


Figure 5.3: Architecture of the autoencoder used in the reconstructing decoders.

get removed from the training set. Then, we reshape all of our data with the autoencoder we defined, just before putting everything into our one-class algorithms. Then, we train OC-SVM, IF, or LOF with the reshaped training set, which means there are 3 different reconstructing detectors. Finally, we introduce the test set to print results. See Figure 5.4 for a better understanding of these detectors.

5.1.4 Neural network detectors

Implementation of the OC-NN and RCAE is quite different than others. It does not involve any ResNet-50, standardization, outlier removal, and PCA. We use the implementation of OC-NN and RCAE [2] by Chalapathy et al.[10] that runs on the MNIST data set[25], which is a database of handwritten digits. We modify it to be able to import our data set. More specifically, we adjust their autoencoder. Because the implementation of them allows 28x28 pixel gray-scale images only, we need to upgrade to 64x64 pixels and exclude monochrome. Higher resolutions were not possible due to our insufficient computational power.

The autoencoder architecture for 64x64 images is introduced in Figure 5.5. Also, to be able to use such an autoencoder, the images inside of the training and the test sets have to be reshaped into 64x64 pixels. These detectors process the input while loading them at the beginning in order to achieve such reshaping. On top of that, we implemented a 32x32 pixel version of this autoencoder to see the outcome of different resolutions. There are only three detectors built under this category. The first uses RCAE and the rest uses OC-NN with different resolution settings.

[2] `https://github.com/raghavchalapathy/oc-nn`

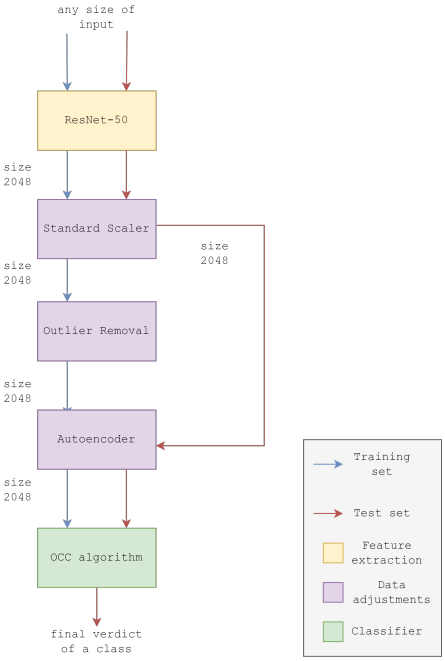


Figure 5.4: Architecture of the reconstructing detectors. ResNet-50 extracts the features of the given input. Then, the standard scaler normalizes the output. Training set gets sanitized by Algorithm 1. Later on, the autoencoder introduced in Figure 5.3 rebuilds all of the previously processed data. Now, an OCC algorithm, IF, OC-SVM, or LOF, trains on the adjusted training set. Finally, the OCC algorithm receives the reconstructed test set and outputs predictions.

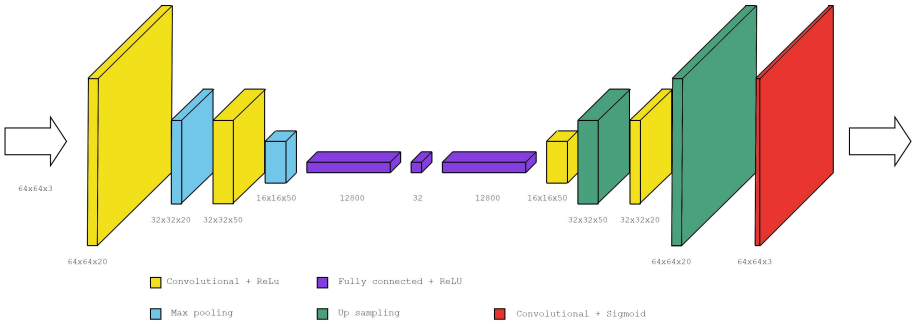


Figure 5.5: Autoencoder used in RCAE and OC-NN, 64x64x3 input suitable version built for this research.

5.2 Environments

This section explains the two main setups that we run our detectors. By implementing these test environments, we would like to have a conclusion on the research questions that are proposed in this paper.

5.2.1 Feature extraction-based environment

This first experiment environment is completely built by us. It aims to run one-class classification algorithms with the combination of feature extraction in order to distinguish if given images are human-made or not. As a result, we expect to see the accuracy metrics of the different detectors, which run in this environment. In addition to that, we want to be able to analyze the metadata collected on the results. Thus, we may improve the detectors or adjust the data set in order to increase the accuracy of the overall experiments in the future.

This environment includes every standard detector, PCA detector, and reconstructing detector. We run each of these 9 different detectors in total on 10 different seeds. Depending on the seed used, we get different permutations when we split the WikiArt data set into two pieces. Making 10 different splits makes it possible to see the overall success of our detectors. By calculating their average score and standard deviation across 10 seeds, we aim to provide more insights into the performance of these models. We find 10 seeds sufficient enough and computationally friendly for this purpose.

The first split piece, the 90% of the WikiArt set, is considered to be our training set. And the other piece, the 10% of the WikiArt set, adds itself to the OpenArt set in order to be used in testing. See Table 5.1 for a better understanding of split. After the training and the test set preparation phase are done, detectors do their work. The final results we get contain F1-score, accuracy, precision, recall, and AUC score. Moreover, the environment allows for metadata analysis in the end. Therefore, we are able to see two types of analysis of results, which are score-based and metadata related.

5.2.2 Neural network-based environment

The second experiment environment is partially built by us, most of them are built by Chalapathy et al.[10]. This environment aims to get the accuracy metrics from RCAE and OC-NN on the task of distinguishing human-made artworks from AI-made ones. Due to the complex nature of this environment, it was not feasible for us to implement metadata analysis for the outcome of these detectors, which run in this environment.

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

# of unique	images	artists	styles
WikiArt set	81444	1119	27
OpenArt set	130082	50	unknown
Training set	73299	≈1116	≈27
Test set	138227	≈1020	≈27

Table 5.1: Information on the images that belong to the different data sets gathered. The approximate signs for artists and styles state there can be little differences based on the random state used in `train_test_split()` function[4].

The environment includes only neural network detectors, so it only runs RCAE and OC-NN. Like in the first environment, detectors also run on 10 different seeds. Then the same train and test set splitting is applied. However, due to computational capacities, we only take randomly selected 10000 images from each WikiArt and OpenArt set. This randomization is also decided with seeds, so the results are reproducible. After the detectors run, results are produced in terms of AUC score only. We are not able to print out any of the other accuracy metrics due to the complex technicality of the initial environment we started to build on.

5.3 Hyperparameters

Hyperparameters are significantly important, especially with machine learning algorithms. As IF, OC-SVM, and LOF are all machine learning algorithms, the parameters of these algorithms may affect the final result in a positive or a negative way. In this research, we decided to avoid hyperparameter tuning because these parameters are highly dependent on the data sets introduced for the task. Therefore, the results may easily get distracted by a change of sets defined. This distraction can result in a positive or negative effect in the end. That being said, tuning hyperparameters would be completely fine if there were no future work window. But as this research is improvable and human-made artworks are being produced every day more than ever, we find such tuning unnecessary at the moment. So, most of the parameters used in the environments are set to their default values. And some parameters that affect the final outcome visibly are fixated by us in order to provide a reference point for future researchers. Therefore, we do not completely rely on the sklearn authors on this matter. More information can be found in Appendix A.

Chapter 6

Results

In this chapter, we provide the accuracy metrics of our detectors in terms of AUC scores. Following that, metadata statistics are highlighted, which are derived from our best results. These statistics represent the worst and the best-performing top 5 values for each metadata attribute.

6.1 Results in terms of AUC scores

Table 6.1 highlights the performance of the detectors on the mission of distinguishing fake images from real ones. All of the specifications of the detectors and the environments are kept unchanged.

Detector type	Classifier	AUC score
Standard	IF	0.55 (± 0.01)
	OC-SVM	0.56 (±0.0)
	LOF	0.51 (±0.0)
PCA	IF	0.55 (±0.0)
	OC-SVM	0.56 (±0.0)
	LOF	0.50 (±0.0)
Reconstructing	IF	0.54 (±0.01)
	OC-SVM	0.53 (±0.0)
	LOF	**0.57 (±0.0)**
Neural network	RCAE	0.57 (±0.02)
	OC-NN (32x32)	0.55 (±0.05)
	OC-NN (64x64)	0.57 (±0.05)

Table 6.1: Average AUC scores of our 12 detectors over 10 runs, standard deviations are represented in parentheses. The best results are shown in bold.

Table 6.2 shows the accuracy of the standard detectors using different versions of ResNets in the feature extraction process, which implies the specification of the standard detectors is adjusted. In order to evaluate ResNet-50's performance, the 152-layered ResNet and the resampler-implemented version of the ResNet-50 replace ResNet-50.

Table 6.3 represents the performance of the standard detectors using different models for the feature extraction, which refers to the specification of the standard detectors being adjusted. To test the state of the ResNet-50, DenseNet, EfficientNet, and Xception are used to fulfill ResNet-50's duty.

	IF	OC-SVM	LOF
ResNet-50	**0.54**	**0.56**	**0.51**
ResNet-152	0.51	0.54	0.51
ResNetRS-50	0.52	0.53	0.50

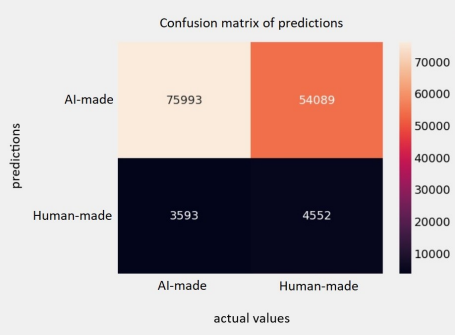
Table 6.2: AUC scores on the test set we defined for this research on a specific random state. Using the standard detectors with different versions of ResNet builds for the feature extraction process on our task to answer RQ-1. The best results are shown in bold.

	IF	OC-SVM	LOF
ResNet-50	**0.54**	**0.56**	0.51
DenseNet-121	0.48	0.48	0.50
EfficientNet-B7	0.50	0.49	0.52
Xception	0.47	0.48	**0.54**

Table 6.3: AUC scores on the test set defined for this research on a specific random state. Using the standard detectors with different feature extraction models on our task to answer RQ-1. The best results are shown in bold.

6.2 Accuracy results

As the reconstructing detector that uses LOF is the best model, we provide additional details of its results. This detector results in an accuracy score of 58% and it has at least 56% precision for both classes in the overall test. See Figure 6.1 for the specifics. However, all of the other versions of LOF (standard and PCA) average an 8% accuracy. And all the other detectors that use IF and OC-SVM score 23% on average in terms of accuracy in the overall test. Furthermore, these unsuccessful LOF detectors predict 98% of the test set is human-made paintings. And the incompetent IF and OC-SVM detectors claim 83% of the test set is human-made artworks. Because of the reasons explained in the earlier chapter, any information does not exist for RCAE and OC-NN in terms of accuracy score.



	precision	recall	f1-score	# images
AI-made	0.95	0.58	0.72	130082
Human-made	0.08	0.56	0.14	8145
overall accuracy			0.58	138227

Figure 6.1: On the left, confusion matrix of the predicted values from the top-performing detector, which is the reconstructing LOF detector. On the right, different assessment metrics are derived from the confusion matrix. Numbers are produced over a single run.

6.3 Metadata analysis

This subsection introduces the analysis of metadata by highlighting the top 5 most easy-to-detect values of different metadata types as well as showing the top 5 most hard-to-detect variables of distinct metadata classes. For instance, in Table 6.4, if an image is drawn using the style of Vincent van Gogh, the model knows if the painting is made by a human or not 87% of the time. Following the same logic, in Table 6.5, if an image is created by the characteristics of Ivan Shishkin, then the model predicts 29% accurate results on if the image is AI-made or not. The rest follows the same idea for all the attributes in these tables. According to the Table 6.1, the top accurate model is the reconstructing detector that uses LOF. Therefore, the statistics given are provided by the predictions of that specific detector.

Values with low occurrence and sharp outcomes are unappreciated. For example, some resolution scales are apparent only once but have an accuracy of 0% or 100%. Therefore, we set a threshold value for each metadata category. These thresholds state the minimum number of images needed for a meaningful accuracy analysis in the end. If the number of images is above the threshold, we consider these values can generalize the behavior of the detector. These numbers are 200 for artists, 45 for styles, 22 for models, 182 for prompt words, 15 for prompt lengths, 15 for negative prompt words, 15 for negative prompt lengths, 14 for guidance scales, 35 for samplers, 20 for resolution scales, and 10 for steps. The numbers are fixed based on our observation of the distribution of the values in our test set.

Top	Artist	Style	Model	Prompt word	Prompt length	Negative prompt word	Negative prompt length	Guidance scale	Sampler	Resolution scale	Steps
1	Vincent van Gogh [0.87]	Color field painting [0.77]	wavymulder/modelshoot [0.7]	photoy[0.99]	63[1.0]	naked[0.78]	47[0.82]	30 [1.0]	DDIM [0.7]	35 x 48 [1.0]	37 [0.93]
2	Gustave Dore [0.87]	Minimalism [0.73]	Stable Diffusion (v2.1) [0.64]	lothian[0.97]	55[0.92]	nude[0.78]	48[0.81]	10 [0.86]	Unknown [0.58]	13 x 14 [1.0]	31 [0.92]
3	Salvador Dali [0.85]	Rococo [0.67]	Openjourney V2 [0.63]	danilo [0.97]	66[0.89]	nudity[0.68]	2[0.70]	13 [0.77]	Eular A [0.49]	15 x 17 [1.0]	200 [0.87]
4	Henri de Toulouse Lautrec [0.78]	Impressionism [0.65]	Stable Diffusion (v1.5) [0.60]	meyers[0.97]	71[0.89]	split[0.5]	46[0.69]	15 [0.67]	DPM Solver++[0.47]	9 x 8 [1.0]	86 [0.73]
5	Raphael Kirchner [0.75]	Pointillism [0.65]	Stable Diffusion [0.59]	reimann [0.97]	65 [0.89]	grainy [0.47]	NaN [0.59]	16 [0.64]		11 x 12 [1.0]	46 [0.73]

Table 6.4: Top 5 easy-to-detect values of each metadata category. If an arbitrary image is created by the specifics given in the table, the reconstructing LOF detector distinguishes if it is AI-made or not by the accuracy shown in square brackets.

Top	Artist	Style	Model	Prompt word	Prompt length	Negative prompt word	Negative prompt length	Guidance scale	Sampler	Resolution scale	Steps
1	Ivan Shishkin [0.29]	Expressionism [0.38]	SG161222/Realistic_Vision_V1.3 [0.09]	anatomical [0.0]	74[0.42]	anime[0.0]	4[0.13]	17 [0.08]	DPM Solver++ [0.47]	10 x 7 [0.02]	28 [0.09]
2	Isaac Levitan [0.37]	Art Nouveau Modern [0.43]	DreamShaper [0.32]	technology [0.0]	62[0.5]	faces[0.0]	3[0.32]	5 [0.11]	Euler A [0.49]	12 x 13 [0.09]	80 [0.20]
3	Camille Corot [0.38]	Contemporary realism [0.43]	Stable Diffusion 2.1 (768) [0.41]	graffiti [0.0]	3[0.51]	picture[0.0]	5[0.33]	9 [0.19]	Unknown [0.58]	11 x 13 [0.1]	71 [0.20]
4	David Burliuk [0.39]	Naive art primitivism [0.45]	DALL-E 2 [0.44] [0.41]	filmic [0.0]	4[0.51]	fat[0.0]	6[0.29]	9 [0.19]	DDIM [0.7]	16 x 197 [0.11]	20 [0.25]
5	Ivan Aivazovsky [0.39]	Symbolism [0.45]	Stable Diffusion 1.5 [0.45]	burrito [0.0]	12[0.52]	cropping [0.0]	75[0.36]	12 [0.42]		38 x 25 [0.13]	79 [0.27]

Table 6.5: Top 5 hard-to-detect values of each metadata category. If an arbitrary image is created by the specifics given in the table, the reconstructing LOF detector distinguishes if it is AI-made or not by the accuracy shown in square brackets.

Chapter 7

Limitations and future work

7.1 Limitations

7.1.1 Computational overload

Using ResNet-50 not only provides a feature extraction but also grants efficient space usage for the rest of the computations. WikiArt data set by itself is 25.6 GBs while the images from the OpenArt data set take 20.5 GBs. After the extraction, these numbers reduce to 651MBs and 2.01GBs respectively, which makes the problem feasible for our computers. So, using a residual network solves the memory problem. But on the other hand, it may cause information loss. Therefore, if we had sufficient memory and computational power, we might skip the feature extraction part. Or at least we would have an opportunity to discuss it here after some testing. That being said, due to using raw inputs, imbalances in the training set might rise. Thus, there is a chance that OCC algorithms may suffer.

On top of this information, the existence of autoencoders also pressures memory usage. Our computers were barely capable to cope with such a working load. Therefore, when running neural network detectors, we had to use only 10000 images per each WikiArt and OpenArt set with 64x64 pixel resolution.

In all of our experiments, we used a computer with 16GB RAM and 8GB VRAM.

7.1.2 Data sets

As it is mentioned earlier, we collected the human-made artworks from the research that imported a part of the WikiArt website, which has around 80000 images. However, the current version of this website stores more than 250000 artworks at the moment. Even though some portion of these pieces are from other genres of art like sculptures and jewelry, we know that our version of WikiArt misses many paintings for certain. Moreover, when we

visit the WikiArt website and look for a specific artwork, we see there is more side-information than what we have right now, such as date, tags, and the location of the artwork. Such metadata information could be handy while we were showing the details in Chapter 6.2. Or could be helpful for the research we discuss in the following subsection 7.2.2. While we find the version of WikiArt used in ArtGAN appropriate for our task, the lack of full access to the whole WikiArt can be considered a drawback in our research (in terms of results of the detectors and metadata analysis).

The lack of metadata variety also applies to our AI-made artwork set as this matter is disclosed in Chapter 3. On the other hand, we would like to point out a website called PromptHero[1]. This website is quite generous when it comes to storing metadata. While having all the different attributes of side information we have, it also includes separated prompt tags aforementioned. On top of that, this website includes different categories of art styles, such as photography, anime, fashion, architecture, landscapes, logos, interior design, and 3D renders for interested researchers. However, this website was quite conservative in terms of letting us scrape it, due to their premium plan option. In the case such a premium plan is not activated in an account, reaching images in an automatic manner, like scraping, is impossible as the website limits the visited images per a specific time period. Therefore, we had to move on to the OpenArt.

7.2 Future work

7.2.1 RCAE and OC-NN

Based on the results we got, neural network detectors produce better results in comparison to most of the detectors we built. For researchers interested in this topic, we suggest trying to optimize the current setup or running these algorithms with a much more powerful setup. Therefore they could train and test from the whole data sets with higher resolutions. Table 6.1 shows the accuracy of the OC-NN algorithm in different resolutions. The statistics support the idea of the higher the resolution, the higher the accuracy is. However, this might not be the case. Nevertheless, these numbers should not be slept on.

7.2.2 Class prediction and CountVectorizer (hybrid approach)

As mentioned earlier, ResNet-50 can be used for multi-class image classification and recognition (the version pre-trained with ImageNet). We propose an approach where the 5-10 objects in the given images are recognized by ResNet-50 or some other model. Then, with an algorithm like `CountVectorizer` from sklearn, all of the predicted values get encoded into

[1] `https://prompthero.com/`

a sequence of integers. This process makes the use of OCC algorithms possible as they are not capable to receive strings as input. At least the sklearn implementations are not capable as far as we know. Then, the one-class classification algorithms can try learning from the encoded training set. Then try to distinguish if a given encoded input is abnormal or not. By implementing this algorithm, we can observe the effect of the relation between objects that occurs in images on the final output. However, there are two things worth the mention. The first thing is that ResNet-50 is terrible at recognizing objects from the artworks, see Figure 7.1. In very few cases, like the second artwork in Figure 7.1, these predictions are partially correct. Secondly, we have already built such a model and it results worse than all of our built detectors. It scores 50% on average in terms of AUC score. However, we think that if the first issue is fixed then the second issue is also fixed. Therefore, such a model might work.

Another approach could be making these detectors hybrid. So, instead of using images only, we can use images and tags or prompts of them together. The issue here is that AI-made images have long prompts in general, whereas human-made artworks have short descriptive names and fewer tags. Therefore, we intentionally gave up on the option of collecting names of human-made artworks. Therefore having an imbalanced and obviously detectable hybrid method would not make much sense. In the case of a neutral image descriptor defined, learning the subjects and the contents of the given images would be usable in a hybrid approach.

Figure 7.1: Random images from WikiArt and OpenArt set. For the left image, ResNet-50 claims 'fur coat', 'swimming trunks', 'sunglass', 'cloak', and 'sunglasses' are existing in the image. For the middle one, ResNet-50 claims 'starfish', 'goldfish', 'lakeside', 'coho', and 'king crab' are apparent in the artwork. For the right image, ResNet-50 states 'comic book', 'book jacket', 'pillow', 'jigsaw puzzle', and 'tray' are located.

Chapter 8

Conclusion

We ran 12 different setups to see if one-class classification algorithms can distinguish human-made images from AI-generated ones. Shortly, the answer to this question is 'yes, they are capable of doing that' as can be observed from the results chapter. Therefore, we have an answer to our first research question (RQ-1). But how successful are they is another question. The worst-performing detector we build has an average AUC score of 0.50, which means it produces false positives as much as true positives. So, its predictive ability is not any better than random guessing. On the other hand, our three of the best detectors have an average of 0.57 in terms of AUC score. One of these detectors is built on our original idea, which consists of combining feature extraction and reconstruction of the image by autoencoders before doing the classification process. And the other detectors are the RCAE and OC-NN algorithms with an arguably high standard deviation. Due to their comparably new releases, the algorithms themselves are not popular as others.

In terms of AUC scores, all of these models look nearly precise as another, not good and not terrible. However, except for the reconstructing LOF model, all of the detectors had an accuracy of 10–20% on average in distinguishing AI-made paintings from human-made paintings correctly. The reconstructing LOF detector provided 58% accurate results and it drifted away from all the other detectors on this matter. Thus, we got a clear image of the performance division between differentiating OCC algorithms. In the end, by implementing diverse detectors and by interpreting their results on our task, we were able to get our second research question (RQ-2) answered as well.

We know these anomaly detection algorithms introduced here are successful in some other tasks, such as detecting fake faces, differentiating digits, or anomalies in a stop sign. Therefore, we can not generalize the claim 'observing images are too difficult for OCC algorithms'. However, when it comes down to complex tasks like ours, these algorithms perform just slightly

better than the random coin flip. Even though we appreciate the human-like intelligence characteristics of these algorithms and find them interesting, we would not recommend these algorithms unless there are no alternatives like multi-class classifiers. That being said, the false positives and false negatives should be well monitored in scenarios where these algorithms are used. To conclude, the general state of one-class classification algorithms stays questionable and incompetent against the other classification methods, which implies there is still a window open for future researchers who are interested in this field.

Bibliography

[1] Zeyang Sha, Zheng Li, Ning Yu, and Yang Zhang. De-fake: Detection and attribution of fake images generated by text-to-image diffusion models. 10 2022.

[2] Eunji Kim and Sungzoon Cho. Exposing fake faces through deep neural networks combining content and trace feature extractors. *IEEE Access*, 9:123493–123503, 2021.

[3] 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.

[4] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[5] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 582–588, Cambridge, MA, USA, 1999. MIT Press.

[6] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[7] Markus Breunig, Peer Kröger, Raymond Ng, and Joerg Sander. Lof: Identifying density-based local outliers. volume 29, pages 93–104, 06 2000.

[8] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Robust, deep and inductive anomaly detection. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML*

PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 10, pages 36–51. Springer, 2017.

[9] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[10] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[14] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[15] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[18] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[19] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[20] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[21] Naeem Seliya, Azadeh Abdollah Zadeh, and Taghi M Khoshgoftaar. A literature review on one-class classification and its potential applications in big data. *Journal of Big Data*, 8(1):1–31, 2021.

[22] Shengyin Li, Vibekananda Dutta, Xin He, and Takafumi Matsumaru. Deep learning based one-class detection system for fake faces generated by gan network. *Sensors*, 22(20):7767, 2022.

[23] Wei Ren Tan, Chee Seng Chan, Hernan Aguirre, and Kiyoshi Tanaka. Improved artgan for conditional synthesis of natural image and artwork. *IEEE Transactions on Image Processing*, 28(1):394–409, 2019.

[24] Edward Loper Bird, Steven and Ewan Klein. *Natural language processing with Python.* " O'Reilly Media, Inc.", 2009.

[25] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Appendix A

Appendix

```
IsolationForest(contamination=0.08, max_features=1.0
                , max_samples=1.0, n_estimators=40)

svm.OneClassSVM(gamma=0.001, kernel='rbf', nu=0.08)

LocalOutlierFactor(n_neighbors=200, novelty=True)
```

Figure A.1: The fixed hyperparameters of IF, OC-SVM, and LOF algorithms used in feature extraction-based environments.